НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
“___”_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп’ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система класифікації аудіо контенту»

Виконав:
студент _IV_курсу, групи__ІП-62__
Валігура Ілля Анатолійович _____
(підпис)

Керівник:
Професор, доктор фізико-математичних наук
Гордієнко Юрій Григорович _____
(підпис)

Консультант з нормоконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____
(підпис)

Рецензент
Доцент, кандидат технічних наук
Писаренко Андрій Володимирович _____
(підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМ. ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки
Напрям підготовки - 6.050103 - «Програмна інженерія»

Затверджую:
Завідувач кафедри
Сергій СТІРЕНКО

«____»_____2020 року

ЗАВДАННЯ

на бакалаврську атестаційну роботу студента

Валігури Іллі Анатолійовича

1. Тема роботи «*Система класифікації аудіо контенту*», керівник проєкту д.ф.-м.н., проф., Юрій ГОРДІЄНКО, затверджена наказом по університету від «__»_____2020р. №_____
2. Термін здачі студентом проєкту *5 червня 2020р.*
3. Вихідні дані до роботи *технічне завдання, теоретичні данні.*
4. Зміст розрахунково-пояснювальної записки: *Огляд систем класифікації аудіо контенту, аналіз засобів для створення системи, розробка системи класифікації, висновки.*
5. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В. П.		

6. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітки
1.	Затвердження теми роботи	*01.09.2019*	
2.	Вивчення та аналіз завдання	*15.12.2019-15.03.2020*	
3.	Написання вступної частини та огляд предметної області	*15.03.2020-25.03.2020*	
4.	Розбір принципу роботи, аналіз переваг та недоліків системи.	*25.03.2020-05.04.2020*	
5.	Дизайн системи та розробка алгоритму роботи	*05.04.2020-15.04.2020*	
6.	Програмна реалізація системи	*15.04.2020-20.05.2020*	
7.	Оформлення документації дипломної роботи	*20.05.2020 - 05.06.2020*	
8.	Передзахист	*06.06.2020*	
9.	Захист	19.06.2020	

Студент _____ Ілля ВАЛІГУРА
(підпис)

Керівник проекту _____ Юрій ГОРДІЄНКО
(підпис)

АНОТАЦІЯ

Робота присвячена вирішенню проблеми жанрової класифікації аудіо файлів. Описані різні методи з області пошуку музичної інформації, способи представлення аудіо даних декількох форматів (mp3, wav, au) у цифровому та графічному вигляді, з подальшим їх аналізом і обробкою для налаштування класифікації. Створено новий набір даних (налічує 17000 аудіозаписів розподілених між 14 жанрами), що може бути задіяний у подібних задачах з предметної області.

Запропонована система класифікації використовує класичні методи машинного навчання і різні види глибинних нейронних мереж. В роботі порівнюються підходи до обробки аудіо та алгоритми їх класифікації, а також застосовується об’єднання моделей в ансамбль для покращення ефективності роботи системи.

Розмір пояснювальної записки – 56 аркушів, містить 22 ілюстрації, 4 додатки.

ABSTRACT

The work is devoted to solving the problem of genre classification of audio files. Various methods in the field of music information retrieval, methods of presenting audio data of several formats (mp3, wav, au) in digital and graphical form, with their subsequent analysis and processing to adjust the classification are described. The new set of data has been created (there are 17,000 audio records distributed between 14 genres). It can be used for similar tasks of the subject`s area.

The proposed classification system uses classical machine learning methods and different types of deep neural networks. The paper compares approaches to audio processing and algorithms for their classification. The models are integrated into an ensemble to improve the efficiency of the system.

Explanatory note size - 56 pages, contains 22 illustrations, 4 applications.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система класифікації аудіо контенту”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	3	
2	А4	ІАЛЦ.467100.001 ВП	Відомість дипломного проекту	1	
3	А4	ІАЛЦ.467100.002 ТЗ	Технічне завдання	3	
4	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	61	
5	А3	ІАЛЦ.467100.004 Д1	Принципова схема	1	
6	А3	ІАЛЦ.467100.005 Д2	Структурна схема класифікації зображень згортковою нейронною мережею	1	
7	А3	ІАЛЦ.467100.006 Д3	Функціональна блок-схема	1	

					ІАЛЦ. 467100.001 ВП		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Валігура І.А.*			*Система класифікації аудіо контенту Відомість дипломного проекту*	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Гордієнко Ю.Г.*					*1*	*1*
						НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62		
Н. Контр.		*Сімоненко В.П.*						
Затверд.								

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система класифікації аудіо контенту”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи класифікації аудіо контенту.

Область застосування: автоматизація обробки аудіо файлів, рекомендаційна система для музичних сервісів.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи класифікації аудіо контенту, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи класифікації аудіо контенту.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, довідники з програмованих логічних інтегральних схем, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. ВИМОГИ ДО РОЗРОБЛЯЄМОГО ПРОДУКТУ

- Розробка засобів роботи з різними форматами вхідних аудіофайлів;
- Можливість пакетної обробки файлів;
- Можливість перенавчання системи;
- Розробка засобів візуалізації результатів моделювання;
- Технічна коректність виконання усіх заданих алгоритмів.

5.2. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

- Python 3
- Python data science toolkit (numpy, pandas, scipy, sklearn, tensorflow, keras)

6. ЕТАПИ РОЗРОБКИ

1.	Вивчення та аналіз завдання	*15.12.2019-15.03.2020*
2.	Написання вступної частини та огляд предметної області	*15.03.2020-25.03.2020*
3.	Розбір принципу роботи, аналіз переваг та недоліків системи.	*25.03.2020-05.04.2020*
4.	Дизайн системи та розробка алгоритму роботи	*05.04.2020-15.04.2020*
5.	Тестування окремих моделей системи	*15.04.2020-30.05.2020*
6.	Допрацювання, налагодження і виправлення помилок	*30.05.2020-4.06.2020*

Пояснювальна записка
до дипломного проекту

на тему: «Система класифікації аудіо контенту»

Київ – 2020

					ІАЛЦ.467800.003 ПЗ			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Федорович В.Р.*			*Пояснювальна записка*	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Роковий О.П.*					*2*	*60*
						НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62		
Н. Контр.		*Сімоненко В.П.*						
Затверд.								

ЗМІСТ

ВСТУП

Актуальність

Музика - найпопулярніша форма мистецтва, її виконують і слухають мільярди людей щодня. Музика, що є комбінацією звуків, визначається як гармонійне накладення хвиль різної довжини і частоти, добре упорядкованих музичними елементами, включаючи ритм, мелодію та гармонію. Тому вона піддається обробці та аналізу, як й інші хвильові коливання.

Експоненційний ріст обчислювальної потужності комп'ютерів та стрімкий розвиток технологій зумовив широке застосування систем та засобів штучного інтелекту в різних галузях науки і техніки. Штучний інтелект стає рушійною силою прогресу та одним з головних технотрендом в нашому все більш цифровому світі. Сьогодні в області досліджень так званого штучного інтелекту значного поширення набули системи пошуку музичної інформації, серед яких актуальними задачами є аналіз та обробка аудіо, зокрема музики.

Разом зі збільшенням кількості музичного онлайн-контенту зростає необхідність і в його організації, одним із способів є класифікація за музичним жанром. Музичний жанр – характеристика музики, що полягає у відмінності певних композиційних та стилістичних ознаках, таких як інструментальна та ритмічна структура, характеристика вокалу виконавця, емоційність, танцювальність та інших факторів.

Часто людина володіє тисячами цифрових музичних творів, однак більшість не мають часу і терпіння, щоб організовувати свої особисті колекції, а для професійних музичних баз даних досі наймають робітників, щоб вручну розмічати музичні активи. Значного поширення набули різноманітні сервіси, додатки та платформи для збереження та організації музики, наприклад Spotify, Soundcloud, iTunes, котрі з їх обсягом аудіофайлів мають бути

зацікавленні в автоматизації процесу обробки. Із швидким ростом кількості музики ручна класифікація композицій стає недоцільною та ресурсоємною. Оскільки якість наявних рішень ще не досягла рівня, достатнього для масового впровадження, дана область досліджень є перспективною. Крім того, натреновані на класифікації жанрів системи та підготовлені для цього дані можна використати як основу для більш широких задач, наприклад пошуку схожих музичних композицій або music summarization.

Мета

В цілому мета бакалаврської роботи – це створення системи класифікації аудіо контенту для покращення процесу жанрової класифікації аудіо файлів. Будуть описані технології представлення аудіо даних з подальшим їх використанням для навчання класичних методів машинного навчання та нейронних мереж, а також їх об’єднання в ансамбль для покращення ефективності всієї системи (досягнення кращих показників, ніж будь-якої окремою моделлю, що використовується в ансамблі).

Для досягнення поставленої мети були поставлені наступні основні задачі:

- Провести аналіз можливих рішень;
- Створити систему;
- Дослідити отримані результати.

РОЗДІЛ 1. ОГЛЯД СИСТЕМ КЛАСИФІКАЦІЇ АУДІО КОНТЕНТУ

В той час, коли для людини іноді достатньо декількох секунд прослуховування, щоб ідентифікувати жанр композиції, мільйони людей досі живуть з некласифікованими бібліотеками цифрової музики, а автоматизація цього процесу на високому рівні залишається складною задачею. Сьогодні задача жанрової класифікації музики добре відома, існує багато досліджень цієї теми, що реалізують різноманітні підходи для вирішення проблеми.

Загалом задача зводиться до представлення музичної інформації у цифровому вигляді, виділення значущих ознак і створення класифікатора. Методи машинного навчання та глибинного навчання з застосуванням штучних нейромереж довели свою ефективність у пошуку паттернів та в задачах класифікації загалом, тому є загальновживаними й для жанрової класифікації музики. Отже, існуючі рішення характеризуються відмінностями на кожному з етапів:

1. Підбір датасету;
2. Спосіб представлення музичної інформації у цифровому вигляді (пошук музичної інформації);
3. Виділення ознак;
4. Вибір класифікатора для моделі.

Більшість рішень [13-19] не працюють безпосередньо з аудіо форматом, а використовують заздалегідь підготовлені (обчислені Spotify/Millon Song Dataset/іншими підготовленими наборами даних) ознаки, способи їх обчислення недоступні, а отже не можуть бути обчислені для аудіо. Це означає, всі рішення носять змагальний характер в рамках одного датасета і ніяк не можуть бути застосовані на сирі необроблені аудіо-файли або доріжки, яких немає в конкретному наборі.

До того ж такі рішення використовують (вручну / за допомогою Spotify

API/ Soundcloud API / MSD) artist-level і album-level способи розмітки жанрів аудіо, усі композиції одного виконавця анотуються згідно до одного основного його жанру, проте у наш час виконавці не установлюють собі рамок для творчості, тобто не обмежуються одним жанром, ні на рівні всієї творчості, ні навіть альбомів. MSD використовує зовсім неправильну класифікацію жанрів на кшталт Pop / Rock, коли Metallica і Spice Girls - одне і те ж (ні).

Також існують рішення, що класифікують за жанром шляхом аналізу тексту пісень з використанням технік Natural Language Processing (NLP) [1] [2], такий підхід виявив високу кореляцію між текстом і жанровою приналежністю, однак досить обмежений для тих пісень, що не мають слів або мають їх в малій кількості. Тому найбільш доречним для вирішення поставленої задачі є використання інших підходів, що базуються на методах пошуку музичної інформації (Music Information Retrieval, MIR) та способах цифрової обробки аудіо, виділення значущих ознак напряму з музики.

Використання мел-частотних кепстральних коефіцієнтів (MFCC) – один з найпопулярніших способів для представлення аудіо у цифровому вигляді. Вони застосовуються як ознаки для навчання різних класичних алгоритмів класифікації, як наприклад Support Vector Machine [3-6], Decision Trees [3-4] , Random Forest [4].

Інші рішення пропонують представлення аудіо у вигляді зображень (мел-спектрограм) та класифікацію з використанням їх візуальних ознак за допомогою згорткових нейромереж (CNN) [5], комбінують візуальні ознаки та акустичні характеристики для тренування різних моделей і збирають їх в ансамбль[4], наприклад AdaBoost та SVM [6], або ж використовують вихід одних моделей в якості входу для інших [7].

Алгоритми обчислення мел-частотних кепстарльних коефіцієнтів і створення спектрограм включають віконне перетворення Фур’є. Взагалі найбільш розповсюдженим способом обробки цифрових сигналів сьогодні -

перетворення Фур'є. Проте такий метод має й деякі недоліки, що призвело до появи нових. Найбільш суттєвим недоліком перетворення Фур'є є усереднення характерних рис по всій тривалості сигналу, що робить неможливим застосування такого методу при необхідності аналізу змін сигналу в часі. Доцільно користуватись сучасним математичним методом вейвлет-перетворення, коли мова йде про аналіз швидкоплинних нестаціонарних сигналів[11].

Найпопулярнішими наборами даних є GTZAN, Million Song Dataset (MSD), Free Music Archive (FMA), однак кожний має свої нюанси та недоліки (подібні до тих що описані раніше та будуть описані в розділі 3.5.1), отже виникають сумніви щодо універсальності алгоритмів. У дослідженні [8] показано досить суттєву відмінність у роботі CNN та LSTM на різних наборах данних.

Також частим недоліком всіх рішень є велика різниця між точністю на навчальних та тестових наборах даних, що є ознакою перенавчання моделей, відсутність валідаційної вибірки.

ВИСНОВКИ ДО РОЗДІЛУ

Проведено аналіз існуючих рішень, структуровано відповідно до описів, визначено переваги та недоліки найпопулярніших підходів, що необхідно бути враховано при створені системи класифікації аудіо-контенту. Коротко охарактеризовано стан проблеми сьогодні, визначено актуальність роботи.

РОЗДІЛ 2. ОПИС СИСТЕМИ

Оскільки вхідні файли є неочищеним «сирим» звуком, для передбачення жанру кожної доріжки потрібно кілька кроків, які відображаються нижче на рисунку 2.1.

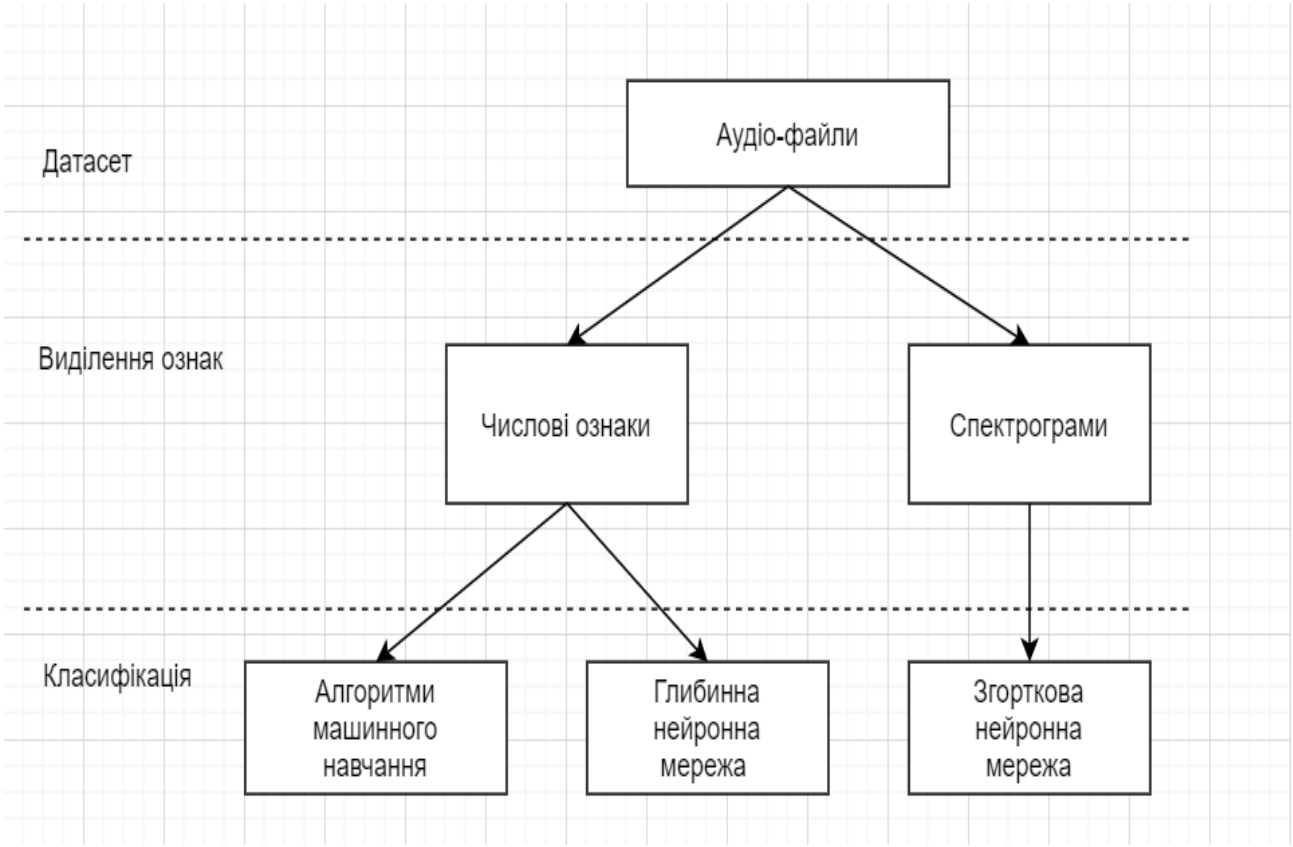


Рис. 2.1 Загальний процес класифікації

По-перше, слід знайти представлення доріжок, яке може бути використане класифікаційними моделями. Наприклад, згорткова нейронна мережа (CNN) може використовуватися разом із графічним представленням звуку (спектрограми), а класичні алгоритми машинного навчання можуть використовувати числові характеристики, такі як мел-частотні кепстральні коефіцієнти, спектральний спад частоти, спектральний центроїд, частоту переходу через нуль та інші . У цьому підході представляється два різних види ознак та різні класифікатори, що можуть їх використовувати, будучи згодом представлені ансамблем для прогнозування жанрів треків. Детальна інформація про кожен крок пояснюється у відповідних підрозділах.

2.1. Набір даних (датасет)

Часто набори даних мають сталі особливості, наприклад тривалість доріжок, частоту дискретизації звуку, бітову швидкість, кількість каналів звуку (моно/стерео). Для того, щоб система була універсальною, тобто могла класифікувати музику незалежно від конкретних особливостей датасетів, доречно використовувати їх комбінацію. Таким чином, було створено набір даних, що поєднує вибірки з GTZAN, Free Music Archive (FMA), власної розміченої бібліотеки.

• GTZAN – Збалансований набір даних, що складається з 1000 аудіозаписів кожна 30 секунд. Він містить 10 жанрів, кожен представлений по 100 треків. Доріжки - це 16-бітні моно-канальні аудіофайли 22050 Гц у форматі .wav.

• Free Music Archive (FMA) – незбалансований набір 100 тис. аудіозаписів, що розділені між 160 жанрами. Доріжки закодовані у форматі mp3, більшість із яких мають частоту дискретизації 44,100 Гц, швидкість передачі бітів 320 кбіт / с (в середньому 263 кбіт / с) та стерео-звук. Дані в архіві не організовані.

• Результуючий набір даних складається з 17000 аудіозаписів, розподілених між 14 жанрами (класична, електронна, хіп-хоп, поп, рок, метал, інструментальна, диско, джаз, блюз, кантрі, фолк, експериментальна, реггі).

Для покращення ефективності роботи системи результуючий набір даних розділяється на три групи, що використовуються на різних етапах створення моделей:

• тренувальний набір (70%). Використовується безпосередньо для навчання моделей (корегування параметрів моделі);

• валідаційний (затверджувальний) набір (20%). Використовується для об'єктивного оцінювання якості моделей, яка відповідає навчальному набору при налаштуванні параметрів та гіперпараметрів (наприклад, число дерев у

алгоритмі Random Forest). Також застовується для регуляризації навчання моделей шляхом ранньої зупинки: навчання переривається за ознакою перенавчання (коли помилка на наборі даних для валідації починає збільшуватись);

• тестовий набір (10%). Приклади не перекриваються з тренувальною вибіркою. Використовується для об'єктивного оцінювання якості кінцевої моделі.

Розподіл відбувається випадковим чином, але шляхом стратифікованих проб. Тобто розподіл по жанрам у кожному з наборів відповідає такому розподілу цілого набору.

2.2. Виділення ознак

Музичний звук має безліч характеристик. У його складі є основний тон, як правило, з максимальною амплітудою, і супутні гармоніки - обертони, найбільш значущими є перші кілька гармонік. Однак спектр являє собою великий набір даних, класична частота дискретизації становить 44100 Гц - на кожну секунду аудіо зберігається 44100 значень, а в стерео - вдвічі більше. Це означає, що 3-хвилинна стерео-пісня містить близько 8,000,000 зразків. Це занадто великий об’єм інформації, що недоцільно використовувати в початковому вигляді для вирішення задачі класифікації. Для того щоб зменшити кількість даних до більш керованого рівня для початку було відкинуто стереоканал, оскільки він містить дуже зайву інформацію. Проте й цього недостатньо, необхідно визначити значущий набір ознак. Процес вилучення інформації для аналізу називається виділенням ознак (feature extraction).

2.2.1 Виділення числових ознак

Найбільш широко використовуваним методом обробки цифрових

сигналів наразі є перетворення Фур'є. Однак воно має ряд недоліків, які привели до пошуку нових удосконалених методів цифрового аналізу нестаціонарних сигналів. Найбільшим недоліком перетворення Фур'є можна назвати усереднення характерних рис по всій тривалості сигналу, що робить неможливим застосування даного методу при необхідності аналізу змін сигналу в часі. Тому задля можливості використання деяких ознак, що обчислюються перетвореннями Фур'є, серед яких мел-частотні кепстральні коефіцієнти та спектральний центроїд, такий недолік було усунуто шляхом розбиття доріжок на сегменти (фрейми, частини), аналізуючи кожен з них окремо.

Отже, виділено такі числові ознаки:

- *Мел-частотні кепстральні коефіцієнти (MFCC)*

Дані коефіцієнти широко використовуються в області пошуку музичної інформації, були визначені як кращі ознаки для розпізнавання музичних інструментів в роботі [9]. Широке застосування саме цих ознак спричинено тим, що вони апроксимують систему слуху людського вуха завдяки зглажуванню різкого сприйняття зміни у нижньому діапазоні частот та навпаки підсиленні сприйняття до змін високочастотних сигналів;

Алгоритм обчислення MFCC можна описати наступним чином [10]:

1) обчислення віконного перетворення Фур'є;
2) нелінійне розбиття спектра на n частин із застосуванням мел-шкали;
3) обчислення енергії сигналу для кожного інтервалу із застосуванням трикутних фільтрів (з перекриттям);
4) обчислення логарифма енергії сигналу для кожного інтервалу;
5) виконання дискретного косинусного перетворення.

- *Спектральний центроїд*

Спектральний центр ваги вказує, де розташований "центр мас" звуку, розраховується як середньозважене значення всіх частот. У блюзових

композиціях частоти рівномірно розподілені, і центр ваги лежить десь в середині спектра. У металі спостерігається виражене зміщення частот до кінця композиції, тому і спектроїд лежить ближче до кінця спектра;

- *Частота перетину нуля*

Частота перетину нуля - це частота зміни знаку сигналу, тобто частота, з якою сигнал змінюється з позитивного на негативний і навпаки. Ця функція широко використовується як для розпізнавання мови, так і для вилучення музичної інформації. Для металу і року цей параметр зазвичай вище, ніж для інших жанрів, через велику кількість ударних;

- *Частота кольоровості*

Кольоровість (chroma features) - це цікаве і потужне уявлення для музичного звуку, при якому весь спектр проектується на 12 контейнерів, які представляють 12 різних півтонів музичної октави;

- *Спектральний спад частоти*

Спектральний спад частоти - міра форми сигналу, що представляє собою частоту, нижче якої лежить певний відсоток від загальної спектральної потужності, наприклад, 85%.

Розмірність отриманого простору ознак після застосування моментів буде сягати близько 500. Тому доцільно застосувати метод головних компонент (PCA). Це дозволило б виділити найбільш значущі ознаки серед усіх, а також позбавитись тих, що несуть надлишкову інформацію (шкідливу для тренування класифікаторів). Така надлишковість може виникати через кореляцію деяких ознак.

2.2.2 Виділення візуальних ознак

Спектрограма - це візуальне уявлення спектра частот звукових або інших сигналів, що змінюються з часом. Іноді їх також називають сонограми. Це зображення показує залежність спектральної щільності потужності сигналу

від часу. Найбільш поширеним уявленням спектрограми є двовимірна діаграма: на горизонтальній осі представлено час, по вертикальній осі - частота; третій вимір із зазначенням амплітуди на певній частоті в конкретний момент часу представлено інтенсивністю або кольором кожної точки зображення. Спектрограма зазвичай створюється одним з двох способів: апроксимується, як набір фільтрів, отриманих із серії смугових фільтрів (це був єдиний спосіб до появи сучасних методів цифрової обробки сигналів), або розраховується за сигналом часу, використовуючи віконне перетворення Фур'є.

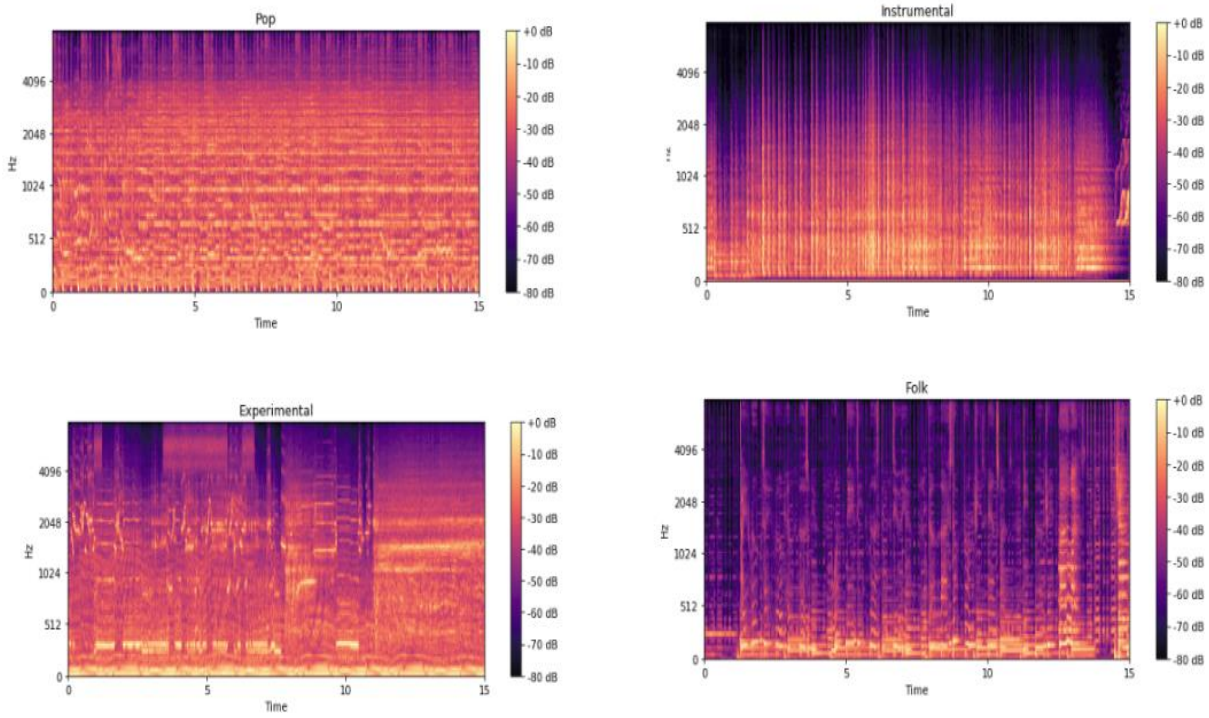


Рис. 2.2 Спектрограми доріжок різних жанрів

Як видно на рис 2.2., спектограми різних музичних жанрів суттєво відрізняються, що додає впевненості в доцільності застосування згорткових нейронних мереж. Оскільки всі значущі зміни відбуваються в нижній частині спектра, частотну вісь можна перетворити в логарифмічну.

2.3 Класифікація

2.3.1 Випадковий ліс (Random Forest)

Дерево рішень - інтуїтивно зрозуміла базова одиниця алгоритму випадкового лісу (Random Forest). Його можна розглядати як серію питань так / ні до вхідних даних. В кінцевому підсумку питання призводять до передбачення певного класу (або величини в разі регресії). Це модель, що легко інтерпретується, так як рішення приймаються так само, як і людиною: ми задаємо питання про доступних даних до тих пір, поки не дійдемо до певного рішення (в ідеальному світі).

Базова ідея дерева рішень полягає у формуванні запитів, з якими алгоритм звертається до даних. При використанні алгоритму CART питання (поділ вузлів) визначаються таким чином, щоб відповіді вели до зменшення забруднення Джині. Це означає, що дерево рішень формує вузли, що містять велику кількість зразків (з набору вихідних даних), що належать до одного класу. Алгоритм намагається виявити параметри з подібними значеннями.

Забруднення Джині - ймовірність невірної маркування в вузлі випадково обраного зразка. У конкретному вузлі ймовірність невірної класифікації зразка можна обчислити за допомогою рівняння:

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2 \qquad (1)$$

тобто забруднення Джині вузла n дорівнює 1 мінус сума pi (частот представників різних класів в листі дерева), зведених в квадрат, для кожного з класів J.

У кожному вузлі дерево рішень шукає такі значення параметрів, що призводять до максимального зменшення забруднення Джині. Потім процес поділу повторюється з використанням «жадібної» рекурсивної процедури, поки дерево не досягне максимальної глибини або в кожному вузлі не

залишаться тільки зразки одного класу.

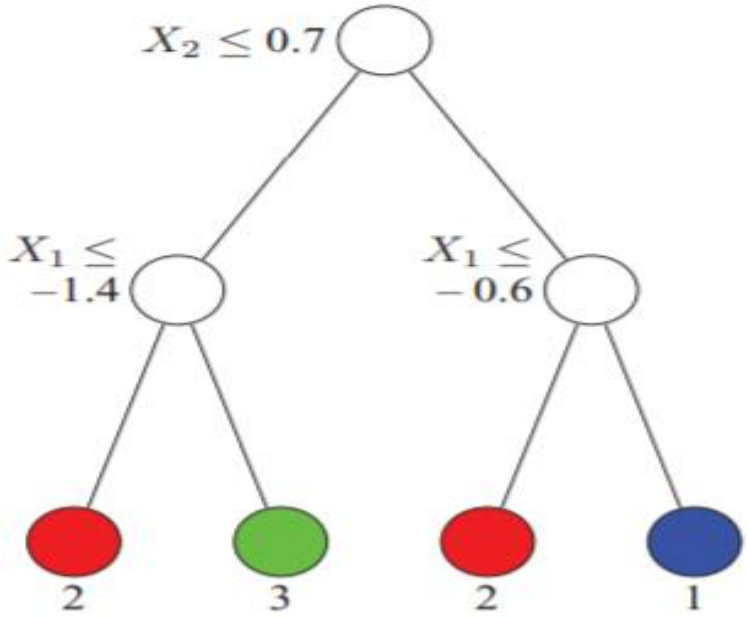


Рисунок 2.3. Структура дерева рішень, для класифікаційної моделі з класами, позначеними як 1, 2, 3

Забруднення Джині має зменшуватися з кожним рівнем, а значення останнього рівня зводиться до нуля. Це означає, що кожен кінцевий вузол містить тільки зразки одного класу, і випадково обраний зразок не може бути невірно класифікований.

Приклад роботи алгоритму зображено на рис. 2.4.

Ця модель не робить помилок, оскільки необмежена кількість рівнів (глибина) дерева дозволяє моделі «запам’ятати» навчальний набір даних, підігнавши вузли під нього. Проте це алгоритм не здатен узагальнювати і правильно обробляти нові дані, тобто відбувається перенавчання, модель стає занадто гнучкою. З іншого боку, у недостатньо гнучкою моделі буде високий рівень похибки, оскільки вона робить припущення щодо тренувальних даних (модель зміщується в бік упереджених припущень про дані), виявляється недостатньо ємною навіть для відповідності тренувальним даним. Пошук балансу між зайвою і недостатньою гнучкістю моделі є ключовою концепцією

машинного навчання і називається компромісом між варіативністю і похибкою (bias-variance tradeoff).

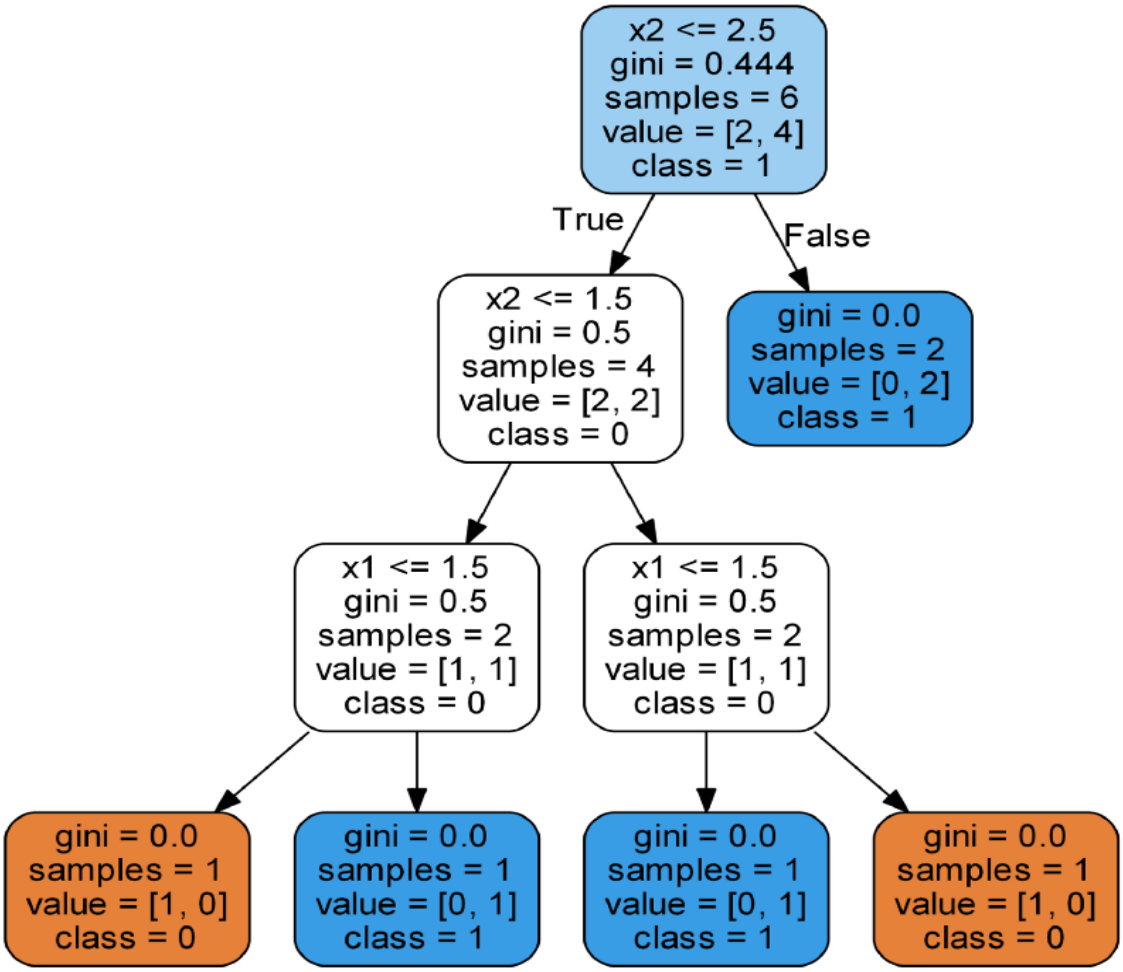


Рисунок 2.4. Візуалізація роботи моделі «Дерево рішень»

Тому для вирішення завдання не вистачить одного дерева рішень. В якості альтернативи обмеження глибини, яке веде до зменшення варіативності і збільшення похибки, ми можемо зібрати безліч дерев в єдину модель. Це і буде класифікатор на основі комітету дерев прийняття рішень або просто «випадковий ліс».

Випадковий ліс - модель, що складається з безлічі дерев рішень. Замість того, щоб просто усереднювати прогнози різних дерев (така концепція називається просто «ліс»), ця модель використовує дві ключові концепції, які і роблять цей ліс випадковим:

- *Випадкова вибірка зразків з набору даних при побудові дерев*

В процесі тренування кожне дерево випадкового лісу вчиться на випадковому зразку з набору даних (бутстреппінг). Хоча кожне дерево може бути високоваріативним по відношенню до певного набору тренувальних даних, навчання дерев на різних наборах зразків дозволяє знизити загальну варіативність лісу, не жертвуючи точністю. При тестуванні результат виводиться шляхом усереднення прогнозів, отриманих від кожного дерева(беггінг);

- *При поділі вузлів вибираються випадкові набори параметрів.*

Використання певної вибірки параметрів зразка для поділу кожного вузла в кожному окремому дереві. Зазвичай розмір вибірки дорівнює квадратному кореню із загального числа параметрів, але цей параметр моделі потрібно налаштовувати.

Отже, випадковий ліс поєднує сотні або тисячі дерев прийняття рішень, навчаючи кожне на окремому вибірці даних, розділяючи вузли в кожному дереві з використанням обмеженого набору параметрів. Підсумковий прогноз робиться шляхом усереднення прогнозів від всіх дерев.

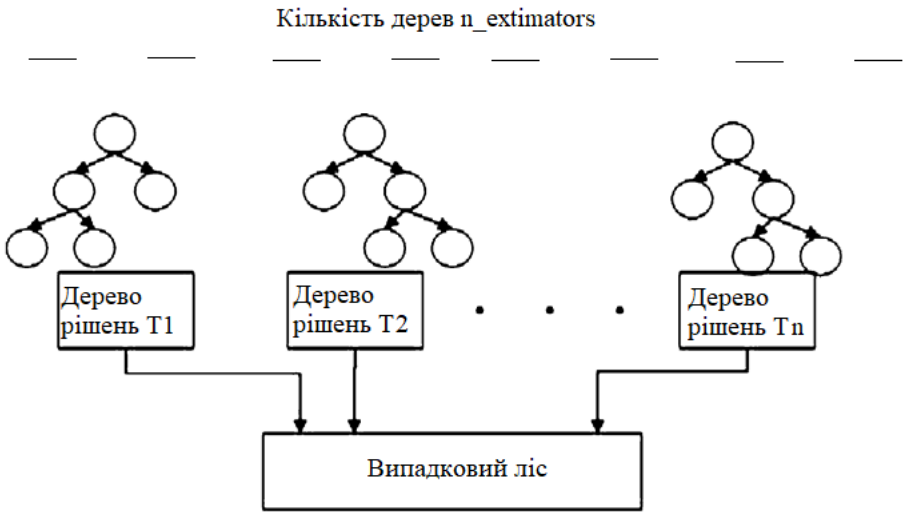


Рис. 2.5 Робота алгоритму випадкового лісу

При налаштуванні алгоритму випадкового лісу зазвичай підбираються такі гіперпарамтри:

1) Кількість дерев n_estimators:

Чим більше дерев, тим краща якість алгоритму на навчальному наборі даних, але час роботи пропорційно збільшується. Після певної кількості дерев у лісі якість алгоритму на тестовому наборі перестане збільшуватись, таким чином можна визначити їх оптимальну кількість;

2) Кількість ознак для вибору при поділі max_features:

Графік залежності якості на тестовому наборі від значення цього параметра унімодальний, отже при інших сталих параметрах легко визначити найбільш підходяще значення. При збільшенні параметра збільшується й час налаштування лісу, а дерева стають більш однообразними;

3) Мінімальна кількість об’єктів для того щоб відбувся поділ min_samples_split;

4) Максимальна глибина дерев max_depth.

2.3.2 Feed-Forward Deep Neural Network (DNN)

Ідеї штучних нейронних мереж сягають 1940-х років. Базова концепція полягає в тому, що мережа штучних нейронів, побудованих з взаємопов'язаних порогових перемикачів, може навчитися розпізнавати паттерни так само, як це робить мозок і нервова система тварин, зокрема, сітківка ока.

Навчання в глибоких нейронних мережах відбувається шляхом закріплення зв'язку між двома нейронами, коли обидва вони одночасно активні під час навчання. У сучасному ПО нейромереж це найчастіше реалізується шляхом змінення значення ваги зв'язків між нейронами, використовуючи правило, зване зворотним поширенням помилки (backpropagation), який більш детально розглянуто пізніше у цьому підрозділі.

Моделюються нейрони таким чином: кожен з них має функцію

поширення, яка перетворює виходи пов'язаних нейронів. Вихід функції поширення переходить в функцію активації, яка спрацьовує, коли її вхід перевищує порогове значення.

У 1940-х і 1950-х роках штучні нейрони використовували функцію ступінчастою активації і називалися персептроном. Про сучасні нейронні мережі можна сказати, що вони використовують персептрони, але насправді вони використовують гладкі функції активації, такі як логістична функція, або сигмоїда, гіперболічний тангенс і лінійний випрямляч (ReLU).

Вихідні значення функцій активації можуть передавати вихідний функції з метою додаткових перетворень. Однак часто вихідний функцією є функція ідентичності, це означає, що вихідний сигнал функції активації передається далі сполученим нейронам.

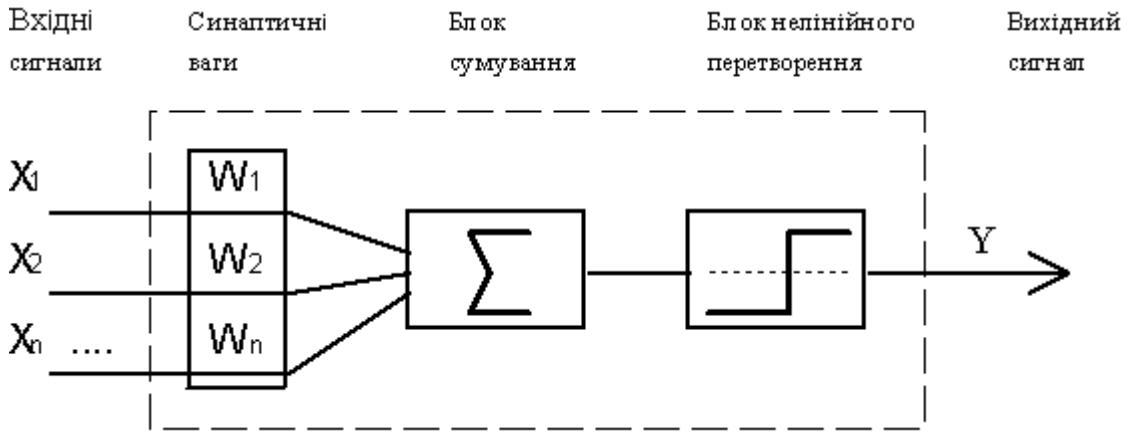


Рис. 2.6 Модель нейрона (персептрона)

Функції активації нейронів:

1) Сигмовидна нелінійна функція має вигляд:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \qquad (2)$$

, показана на Рис. 2.7 (а). Вона приймає на вхід дійсне число і стискає його в діапазон від 0 до 1. Таким чином великі негативні числа стають 0 і великі позитивні числа стають 1. Сигмовидна функція часто

використовувалась протягом довгого часу, так як вона має хорошу інтерпретацію як збудження нейрона: від спокійного стану (0) до повністю насиченого збудження при максимальному значення (1). Зараз сигмовидна нелінійна функція рідко використовується на практиці.

2) Тангенціальна функція:

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \qquad (3)$$

стискає дійсне число і в діапазон від -1 до 1. (Рис. 2.7 (б)). На відміну від сигмовидної функції вихід тангенціальної дорівнює нуль в центрі. Тому на практиці така функція завжди краще сигмовидної нелінійності.

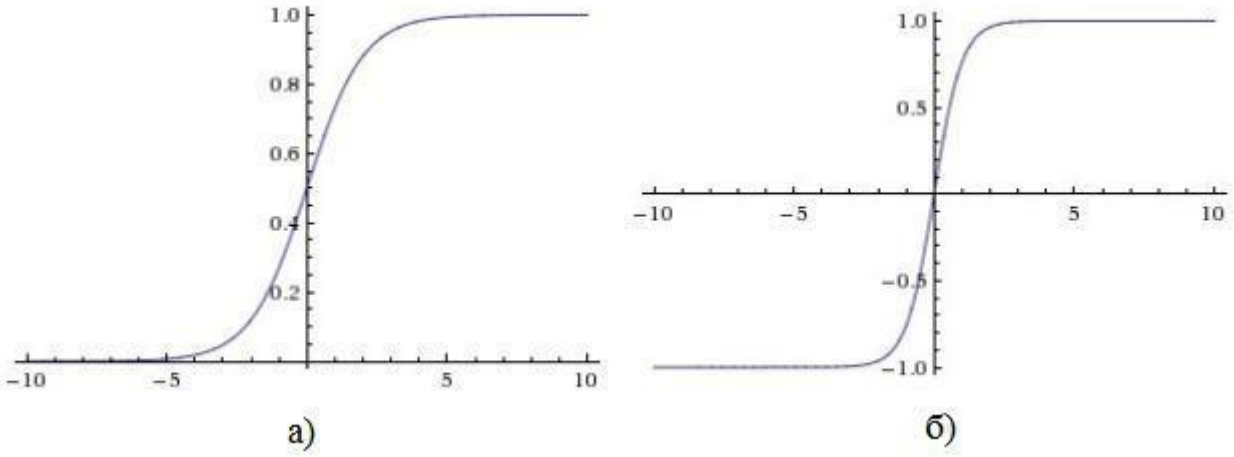


Рис. 2.7 Сигмоїдна(а) та тангенціальна(б) функції активації

3) ReLU, що є найпопулярнішою функцією активації і визначена таким чином:

$$f(x) = \max(0, x) \qquad (4)$$

,тобто функція повертає x, якщо x > 0 і 0 в іншому випадку.

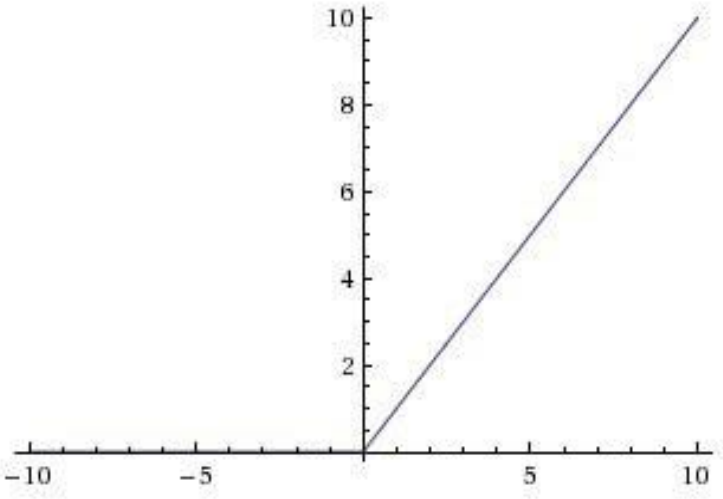


Рис. 2.8 Схема активації нейронів RELU

ReLU стала дуже популярною в останні кілька років. Вона обчислює функцію F (X) = max (0, x). Іншими словами, активація відбувається по нульовому порогу (Рис. 2.8).

Є кілька переваг та недоліків використання ReLUs:

- значне прискорення збіжності стохастичного градієнтного спуску у порівнянні з тангенціальною та сигмовидною функціями;

- в порівнянні з тангенціальною та сигмовидною функціями, які включають дорогі операції (експонент і т.д.), ReLU може бути реалізований за допомогою простої порогової матриці активацій в нулі.

- великий градієнт, що протікає через нейрон ReLU може оновити ваги таким чином, що нейрон ніколи не буде активувати знову. Тобто, блоки ReLU можуть незворотньо померти під час тренування. При правильному налаштуванні швидкості навчання це не є частою проблемою.

4) ELU – модифікація RELU:

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases} \quad (5)$$

ELU дуже схожий на RELU, за винятком негативних входів. Вони

обидва знаходяться у формі функції ідентичності для негативних входів. З іншого боку, ELU стає повільним, поки його вихід не дорівнює -α, тоді як RELU різко згладжується.

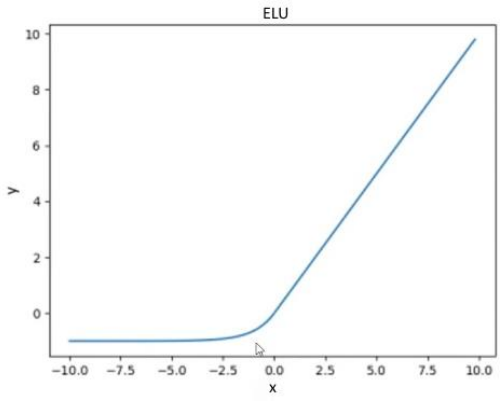


Рис. 2.9 Функція активації нейронів ELU

Приклади топологій нейронних мереж:

1. У мережі прямого зв'язку нейрони організовані в окремі шари: один вхідний шар, будь-яку кількість прихованих шарів обробки і один вихідний шар, а виходи з кожного шару йдуть тільки на наступний шар.

2. У мережі прямого зв'язку з короткими сполуками деякі сполуки можуть перестрибувати через один або кілька проміжних рівнів. У рекурентних нейронних мережах нейрони можуть впливати самі на себе, прямо або побічно, через наступний шар.

Найбільш вдала архітектура нейромережі повинна визначатися дослідницьким шляхом, але вихідний шар для задачі класифікації аудіо-контенту точно міститиме 14 нейронів, що відповідає кількості жанрів у підготовленому датасеті. Функція активації вихідного шару - softmax (або ж нормована експоненційна функція). Softmax - це узагальнення логістичної функції, що "стискує" K-вимірний вектор z із довільним значеннями компонент до K-вимірного вектора sigma (z) з дійсними значеннями компонент в області [0, 1] що в сумі дають одиницю.

$$\sigma : \mathbb{R}^K \to [0,1]^K$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K. \tag{6}$$

«Ймовірності» кожного жанру, замість одного, що має найбільшу, нам необхідні для того, щоб модель мала можливість взяти участь у голосуванні за правильний жанр серед ансамблю моделей.

Модель використовує розріжену категоріальну кросентропію (sparse categorical cross entropy) в якості функції втрат :

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{7}$$

де N – кількість класів, yi – істинне значення класу, yi^ - передбачене мережею

Використання sparse categorical cross entropy обумовлено тим, що мітки передбачуваних класів представлені у вигляді закодованих значень-відповідників (1 – rock, 2 – hiphop і тд), а не у вигляді унітарного коду (one-hot вектору), як того потребує стандартна функція помилки кросентропія (logloss)

Контрольоване навчання нейронної мережі виконується так само, як і будь-який машинне навчання. Береться мережа з групами навчальних даних, порівнюється вихід мережі з бажаним виходом, формується вектор помилок і застосовуються поправки до мережі, грунтуючись на векторі помилок. Пакети навчальних даних, які пропускаються через мережу спільно перед застосуванням поправок, називають епохами.

Метод зворотного поширення помилки (backpropagation) — це ітеративний градієнтний алгоритм, який використовується з метою мінімізації

помилки роботи багатошарового перцептрону та отримання бажаного виходу. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Для можливості застосування методу зворотного поширення помилки функція активації нейронів повинна бути диференційованою.

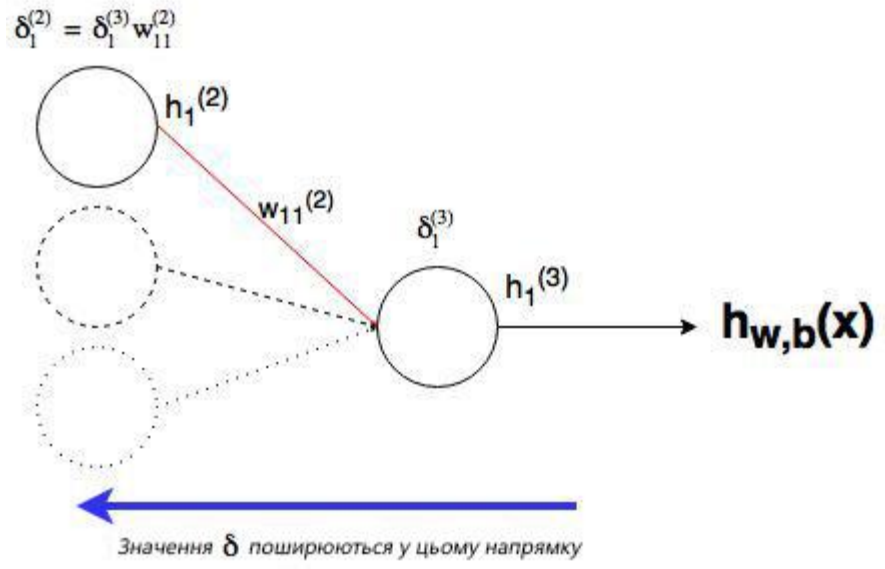


Рис. 2.10 Алгоритм зворотнього поширення помилки backpropagation

Алгоритм використовує градієнт функції помилки (або вартості) щодо ваг і зміщень моделі, щоб виявити правильний напрямок для мінімізації помилки. Застосуванням поправок керують алгоритм оптимізації і змінна швидкості навчання, яка зазвичай повинна бути невеликою, щоб гарантувати відповідність, а функція ReLU не викликала «відмирання» нейронів.

Оптимізатори для нейронних мереж використовують деяку форму алгоритму градієнтного спуску, щоб управляти зворотним поширенням; при цьому часто задіюється механізм, який допомагає уникнути застрявання в локальних мінімумах, таких як оптимізація випадково обраних мініпартій (стохастичний градієнтний спуск), і застосування поправок імпульсу до градієнту. Деякі алгоритми оптимізації також адаптують швидкість навчання параметрів моделі, дивлячись історію градієнтів (AdaGrad, RMSProp і Adam).

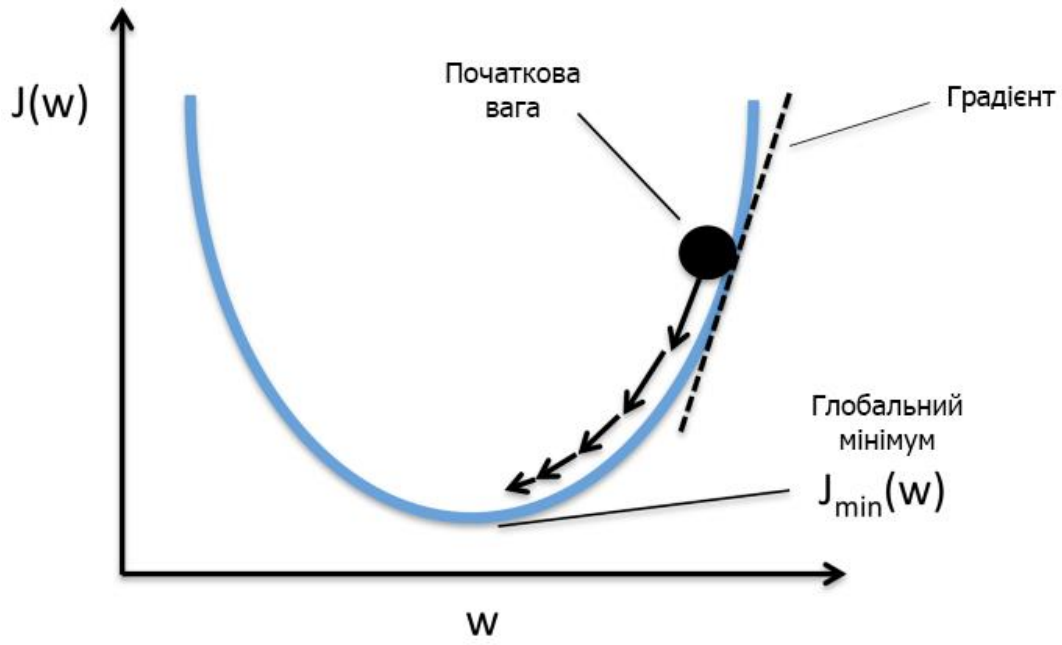


Рис. 2.11 – Демонстрація роботи методу градієнтного спуску

Таким чином, алгоритм діє ітеративно, і його кроки називаються епохами. На кожній епосі на вхід мережі по черзі подаються всі навчальні спостереження, вихідні значення мережі порівнюються з цільовими значеннями і обчислюється помилка. Значення помилки, а також градієнту поверхні помилок використовується для коригування ваг, після чого всі дії повторюються. Початкова конфігурація мережі вибирається випадковим чином, і процес навчання припиняється або коли пройдено певну кількість епох, або коли помилка досягне деякого певного рівня малості, або коли помилка перестане зменшуватися (користувач може сам вибрати необхідна умова зупинки).

На практиці величина кроку береться пропорційної крутизні схилу (так що алгоритм уповільнює хід поблизу мінімуму) з деякою константою, яка називається швидкістю навчання. Правильний вибір швидкості навчання залежить від конкретного завдання і зазвичай здійснюється дослідним шляхом; ця константа може також залежати від часу, зменшуючись у міру просування алгоритму.

Як і в будь-якому машинному навчанні, потрібно перевірити

пророкування нейронної мережі за окремим набору даних перевірки. Без цього є ризик

створити нейронні мережі, які просто запам'ятовують свої вхідні дані замість того, щоб вчитися будувати припущення. Таке відбувається тому, що таким чином ми мінімізуємо не ту помилку, яку насправді необхідно (яку можна очікувати від мережі, коли їй будуть подаватися нові, невідомі раніше, спостереження). Адже нейромережа повинна бути здатною до узагальнення результату на нові спостереження. Але модель навчається мінімізувати помилку на навчальній наборі даних, і за умови відсутності нескінченно великої навчальної множини це зовсім не те ж саме, що мінімізувати "справжню" помилку на поверхні помилок в заздалегідь невідомої моделі явища.

Найсильніше ця відмінність виявляється в проблемі перенавчання, або занадто близькою підгонки, аналогічно до розглянутого явища в попередньому підрозділі при використанні алгоритму Випалкового лісу (Random Forest). Явище перенавчання для нейронної мережі дуже умовно зображено на рисунку 2.12.

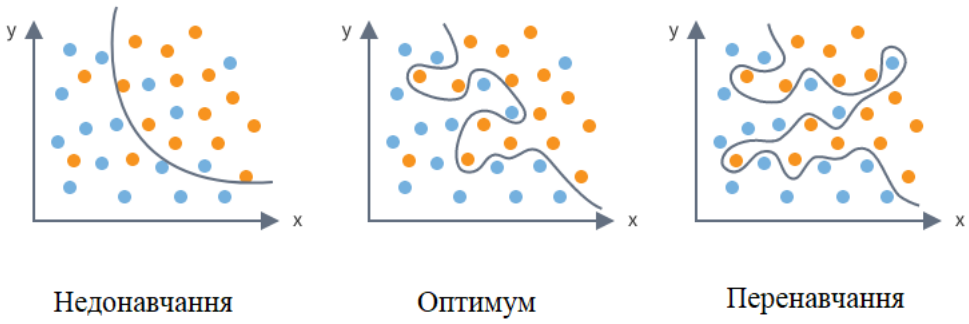


Рис. 2.12 Перенавчання

Мережі з великим числом ваг моделюють більш складні функції і, отже, схильні до перенавчання. Мережа ж з невеликим числом ваг може виявитися

недостатньо гнучкою, щоб змоделювати наявну залежність. Наприклад, мережа без проміжних шарів насправді моделює звичайну лінійну функцію. У більшості випадків більш складна мережа дає меншу помилку, але це може свідчити не про хорошу якість моделі, а про перенавчання.

Виключення або дропаут (англ. dropout) — метод регуляризації штучних нейронних мереж, призначений для запобігання перенавчання мережі. Метод також значно підвищує швидкість навчання. Під час тестування дропаут не застосовується.

Суть методу полягає в тому, що в процесі навчання із загальної мережі випадковим чином виділяється підмережа, для якої здійснюється навчання. Після навчання обраної підмережі випадковим чином обирається нова підмережа і навчання продовжується. Вибір нейронів для підмережі відбувається випадковим чином з ймовірністю, яка називається коефіцієнтом дропаута. Більш навчені нейрони отримують в мережі більшу вагу.

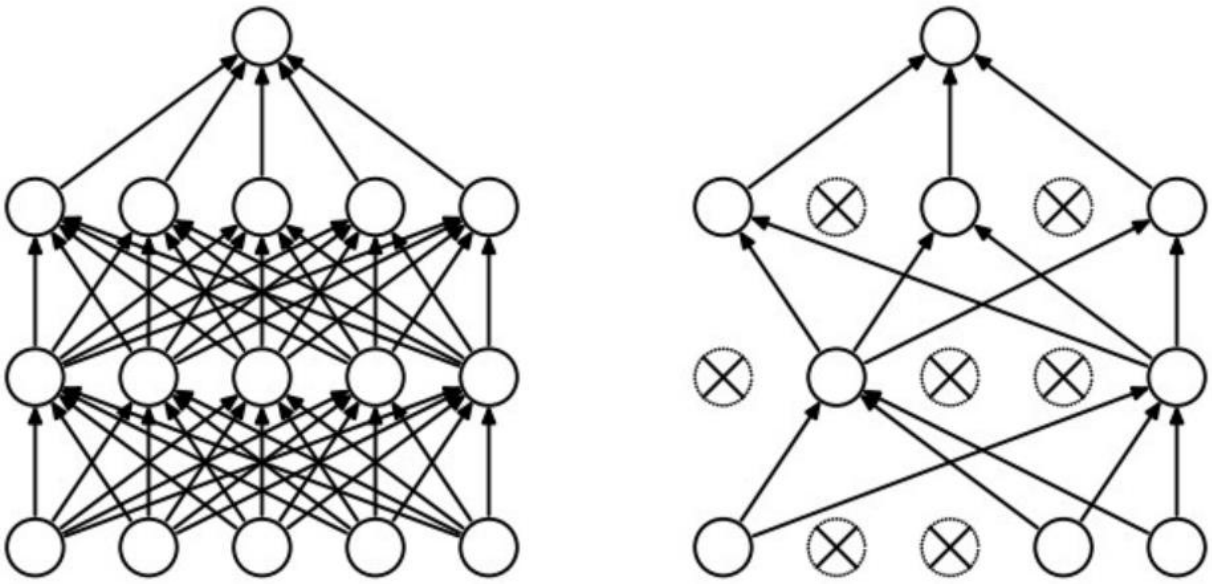


Рис. 2.13 Мережа до(зліва) і після(справа) того як було застовано виключення(дропаут)

Мережі для навчання виходять за допомогою виключення з мережі

(dropping out) нейронів з ймовірністю p, таким чином, вірогідність того, що нейрон залишиться в мережі, становить $q = 1 - p$. "Виключення" нейрона означає, що при будь-яких вхідних даних або параметрах він повертає 0. Виключені нейрони не вносять свій внесок в процес навчання ні на одному з етапів алгоритму зворотного поширення помилки, тому виключення хоча б одного з нейронів рівносильно навчання нової нейронної мережі.

Отже, виділено такі питання, що необхідно вирішити для того, щоб побудувати правильну модель:

1) Топологія і архітектура мережі, кількість шарів на нейронів на кожному з них;
2) Функція активації нейронів;
3) Оптимізатор;
4) Кількість епох та batch розмір;
5) Вірогідність дропауту нейронів;
6) Початкова ініціалізація вагів.

Зазвичай такі рішення приймаються дослідним шляхом, порівняння усіх можливих комбінацій параметрів та гіперпараметрів моделі, а також вибір найбільш вдалих наведено у розділі 3.

2.3.3 Згорткова нейронна мережа

Згорткова нейронна мережа (англ. Convolutional Neural Network, CNN) - це клас глибинних нейронних мереж прямого поширення, який здобув популярність у задачах класифікації аудіо, оскільки показує дуже хороші результати, коли застосовується до візуальних зображень. У випадку з аудіо, як розглядалось раніше, згорткова нейронна мережа може бути використана разом з візуальними представленнями звуку, а саме спектрограмами. Робота CNN загалом дуже схожа до описаного алгоритму роботи звичайної глибинної мережі (згорткова мережа є різновидом багатошарових перцептронів).

Згорткові мережі було натхнено біологічними процесами, в яких схему з'єднання нейронів натхнено організацією зорової кори тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

Згорткова мережа складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації. Цей процес описують як згортку за домовленістю, проте з математичної точки зору він є радше взаємною кореляцією, ніж згорткою.

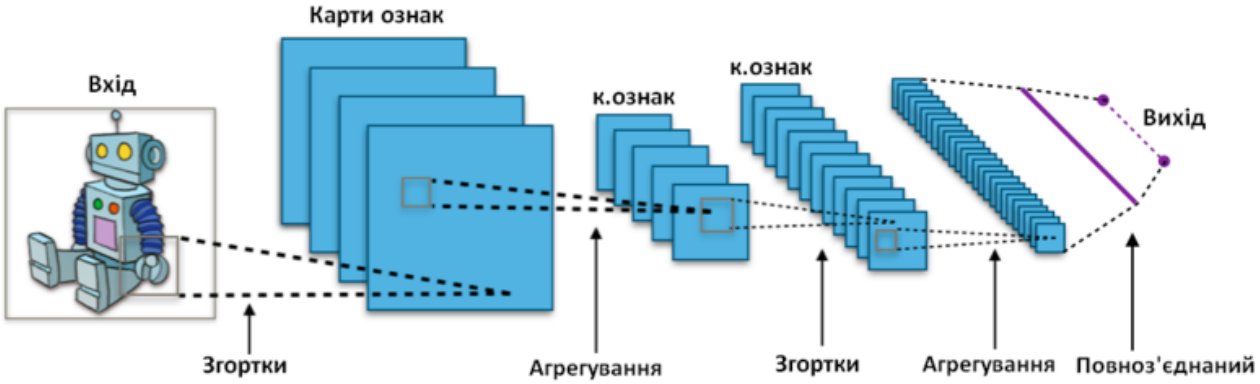


Рис. 2.14 Робота згорткової нейронної мережі

Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул.

Хоч повноз'єднані нейронні мережі прямого поширення й можливо застосовувати як для навчання ознак, так і для класифікування даних, застосування цієї архітектури до зображень є непрактичним. Було би необхідним дуже велике число нейронів, навіть у поверхневій (протилежній до глибинної) архітектурі, через дуже великі розміри входу, пов'язані з зображеннями, де кожен піксель є відповідною змінною. Наприклад, повноз'єднаний шар для (маленького) зображення розміром 100 × 100 має 10 000 ваг. Операція згортки дає змогу розв'язати цю проблему, оскільки вона

зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. Наприклад, незалежно від розміру зображення, області замощування розміру 5 × 5, кожна з одними й тими ж спільними вагами, вимагають лише 25 вільних параметрів. Таким чином, це розв'язує проблему зникання або вибуху градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного поширення.

Розмір ємності виходу згорткового шару контролюють три гіперпараметри:

• Глибина ємності виходу контролює кількість нейронів шару, що з'єднуються з однією й тією ж областю вхідної ємності. Ці нейрони вчаться активуватися для різних ознак входу. Наприклад, якщо перший згортковий шар бере як вхід сире зображення, то різні нейрони вздовж виміру глибини можуть активуватися в присутності різних орієнтованих контурів, або плям кольору.

• Крок контролює те, як стовпчики глибини розподіляються за просторовими вимірами (шириною та висотою). Коли кроком є 1, фільтри

рухаються на один піксель за раз. Це веде до сильного перекриття рецептивних полів між стовпчиками, а також до великих ємностей виходу.

За кроку 2 (або, рідше, 3 чи більше), то фільтри, просуваючись, перестрибують на 2 пікселі за раз. Рецептивні поля перекриваються менше, й отримувана в результаті ємність виходу має менші просторові розміри.

• Іноді зручно доповнювати вхід нулями по краях вхідної ємності. Розмір цього доповнення є третім гіперпараметром. Доповнення забезпечує контроль над просторовим розміром ємності виходу. Зокрема, іноді бажано точно зберігати просторовий розмір вхідної ємності.

Іншим важливим поняттям ЗНМ є агрегування (англ. pooling), яке є різновидом нелінійного зниження дискретизації. Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є

максимізаційне агрегування (англ. max pooling). Воно розділяє вхідне зображення на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак. Агрегувальний шар слугує поступовому скороченню просторового розміру представлення для зменшення кількості параметрів та об'єму обчислень у

мережі, і відтак також для контролю перенавчання. В архітектурі ЗНМ є звичним періодично вставляти агрегувальний шар між послідовними згортковими шарами[9]. Операція агрегування забезпечує ще один різновид інваріантності відносно паралельного перенесення.

Агрегувальний шар діє незалежно на кожен зріз глибини входу, і зменшує його просторовий розмір. Найпоширенішим видом є агрегувальний шар із фільтрами розміру 2×2 (рис. .), що застосовуються з кроком 2, який знижує дискретизацію кожного зрізу глибини входу в 2 рази як за шириною, так і за висотою, відкидаючи 75 % збуджень. В цьому випадку кожна операція взяття максимуму діє над 4 числами. Розмір за глибиною залишається незмінним.

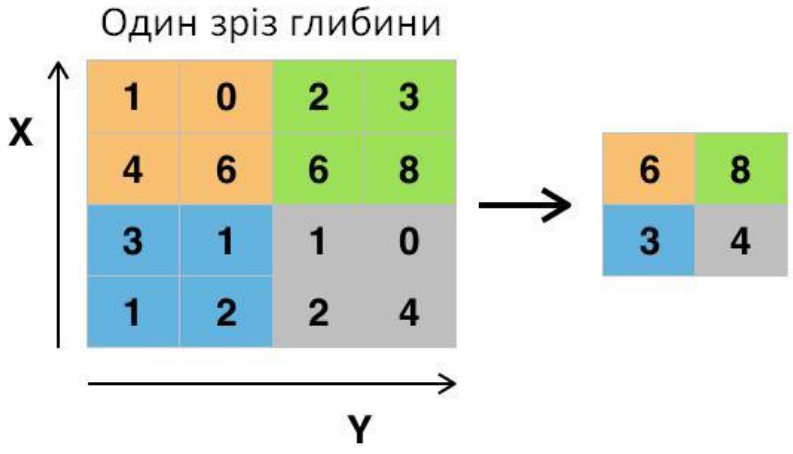


Рис. 2.15 Максимізаційне агрегування (MaxPooling)

Насамкінець, після кількох згорткових та максимізаційно агрегувальних

шарів, високорівневі міркування в нейронній мережі здійснюються повноз'єднаними шарами (англ. fully connected layers). Нейрони у повноз'єднаному шарі мають з'єднання з усіма збудженнями попереднього шару, як це можна бачити у звичайних нейронних мережах. Їхні збудження відтак може бути обчислювано матричним множенням, за яким слідує зсув упередженості. Функція активації повноз'єднаного шару, за аналогією до класичної моделі з попереднього підрозділу, - softmax, згідно до потреби отримати передбачення єдиного класу з K взаємновиключних.

Шар втрат, що визначає, як тренування штрафує відхилення між передбаченими та справжніми мітками, і буде завершальним шаром, буде сигмоїдно перехрестно-ентропійною (sparse cross-entropy loss).

2.3.4 Ансамблеве голосування класифікаторів

Кожен з описаних класифікаторів для кожного спостереження повертає передбачення K (кількість класів) незалежних значень імовірності в проміжку [0;1], сума яких 1. Досягнути кращої ефективності системи можна двома шляхами:

1) Обрати найкращу с точки зору точності класифікації на тестовому наборі даних модель;

2) Створити ансамбль моделей, що будуть приймти рішення щодо класифікації кожного семплу вибірки на основі голосування.

Ансамбль для голосування (або «ансамбль для голосування більшості») - це модель машинного навчання ансамблю, що поєднує в собі прогнози з багатьох різних моделей. Це техніка, яка може бути використана для покращення продуктивності системи, в ідеалі досягнення кращих показників, ніж будь-яка окрема модель, що використовується в ансамблі. У разі класифікації прогнози для кожної моделі підсумовуються та прогнозується мітка класу з більшістю голосів.

Ансамбль для голосування доцільний, коли є дві або більше моделей, які добре справляються з завданням прогнозування моделювання, а також можна припустити, що моделі можуть взаємно доповнювати одна одну (тобто невірно класифіковані дані однієї моделі правильно обробляються іншою). Так як усі класифікатори моделі мають приблизно однакову точність передбачування, було прийнято рішення використати саме голосування, а не одну найкращу модель.

Ансамбль для голосування може вважатися мета-моделлю, зразком моделей. Як мета-модель вона може використовуватися з будь-якою колекцією існуючих тренувальних моделей машинного навчання, а існуючі моделі не повинні знати, що вони використовуються в ансамблі. Це означає, що можна натренувати кожну модель окремо, незалежно одна від одної, а використання ансамблю є опціональним і має на меті лише покращення результату на будь-якому наборі даних.

Існує два підходи до прогнозування більшості голосів для класифікації: жорстке і м'яке голосування. Жорстке голосування передбачає підбиття прогнозів для кожної мітки класу та прогнозування мітки класу з найбільшою кількістю голосів. М'яке голосування передбачає підбиття прогнозованих ймовірностей (або ймовірнісних балів) для кожної мітки класу та прогнозування мітки класу з найбільшою ймовірністю.

Оскільки поділ на жанри аудіо доріжок іноді досить умовний (у випадку поєднання стилів виконавцем), а кількість прогнозованих жанрів суттєво більша за кількість класифікаторів, для кращого перформансу системи краще використовувати м’яке голосування, що передбачено для кожної моделі раніше (кожна з них повертає не мітку жанру, а «впевненість» для кожного жанру в діапазоні [0;1])

Однак проблемою для використання м’якого ансамблевого голосування є деяка неясність в трактуванні прогнозів класифікаторів. До цього часу

вектори, що є результатами роботи окремих моделей інтерпретувались як вектори ймовірностей кожного класу, однак вони є лише певним відображенням «впевененостей» алгоритмів. Спосіб такого наближення може суттєво відрізнятись для різних алгоритмів, а отже для правильного м'якого голосування класифікаторів необхідно попередньо калібрувати їхні такі «імовірнісні бали» до використання в ансамблі.

Калібрування ймовірностей - регулювання розподілу ймовірностей таким чином, щоб краще відповідати очікуваному розподілу, що спостерігається в даних. Ця операція є аналогом масштабування, яка застосовується після того, як передбачення були зроблені за допомогою прогнозної моделі. Існує два популярні підходи до калібрування ймовірностей - платтовий масштаб і ізотонічна регресія.

Оскільки всі класифікатори так чи інакше насправді роблять прогнози лише щодо частини аудіо, а не цілих композицій, голосування необхідно реалізовувати не лише на рівні різних моделей, а й кожної окремо – для класифікації жанру цілого аудіо файлу:

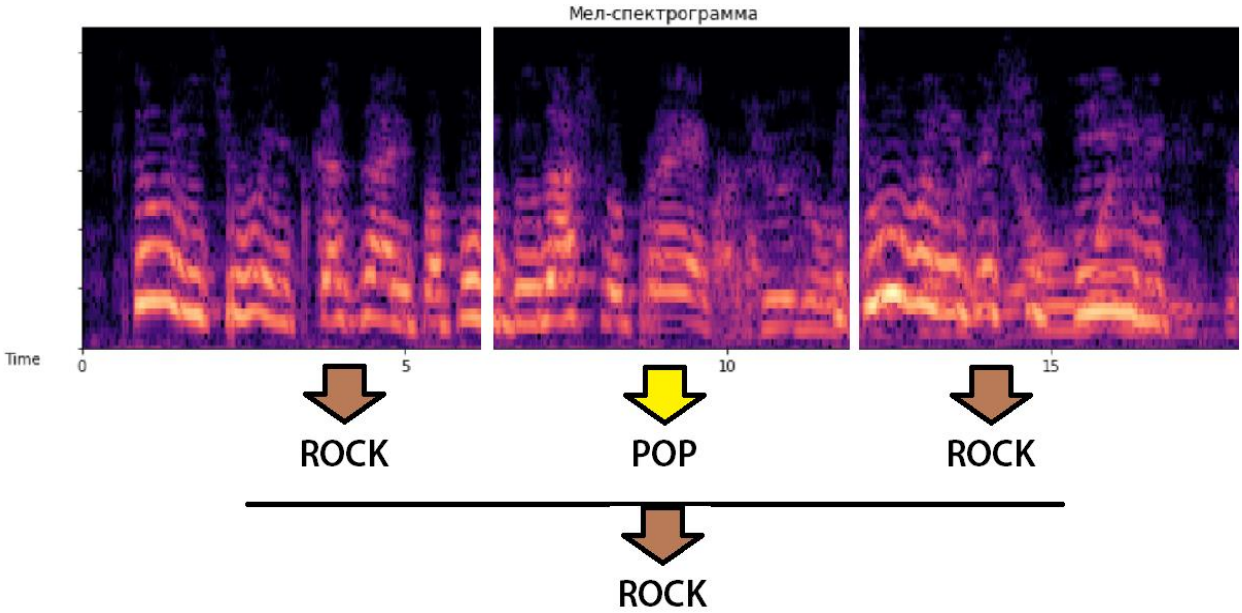


Рис. 2.16 Голосування на рівні одного аудіо файлу, представленого спектрограмою

Таким чином, загальний алгоритм класифікації може бути зображений у вигляді схеми:

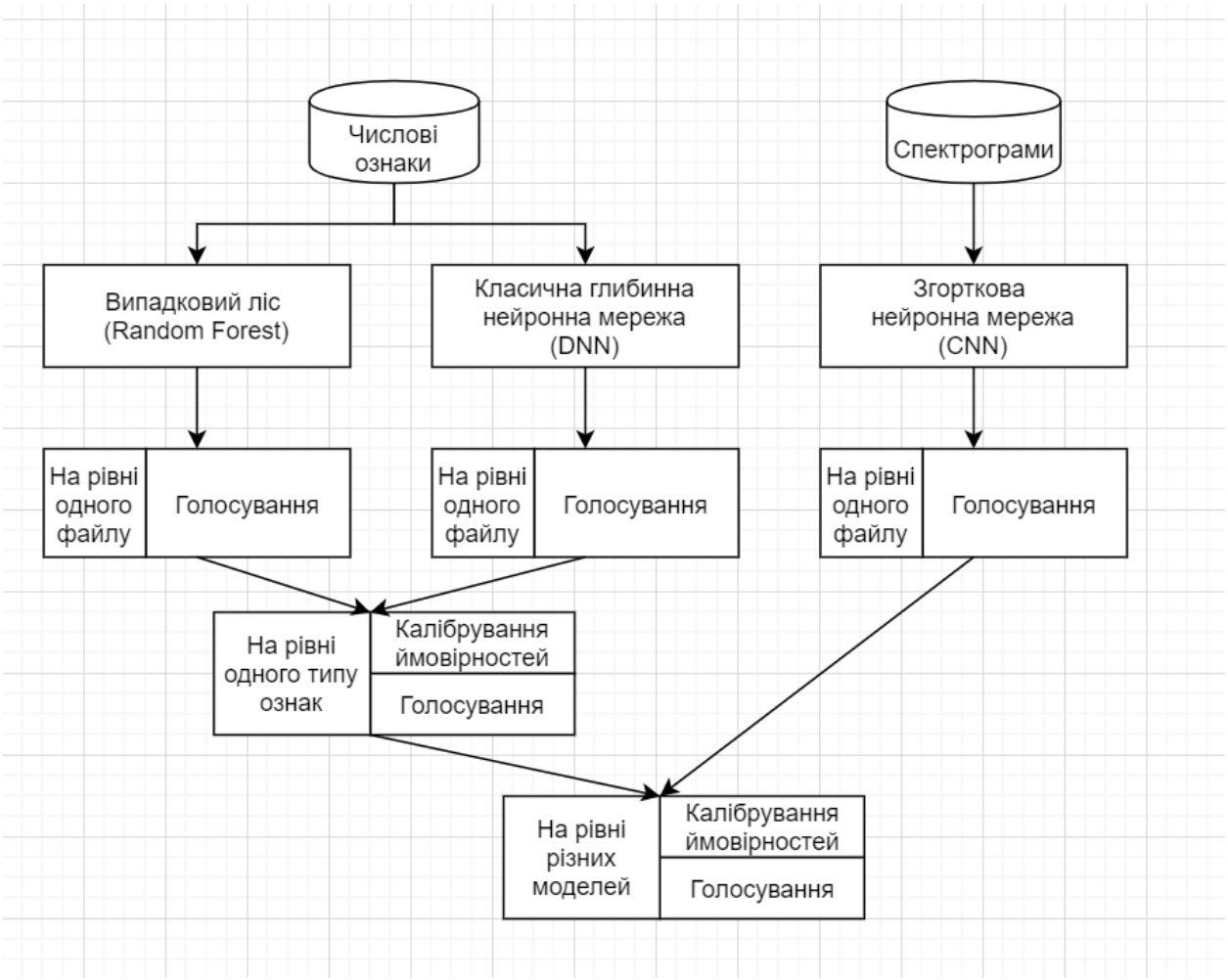


Рис. 2.17 Процес класифікації

ВИСНОВКИ ДО РОЗДІЛУ

Використання моделей Random Forest, Feed-forward Deep Neural Network, Convolutional Neural Network цілком задовольняє вимоги до вирішення задачі класифікації аудіо за жанрами, але метрики якості залежать від правильної конфігурації нейронної мереж. Усі необхідні математичні моделі класифікаторів та сучасні підходи, що будуть використані у системі, описано теоретично.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

Система класифікації аудіо-контенту в режимі реального часу не повинна витрачати час на створення та підготовку набору данних для тренування, виділення найкращих ознак, компонування моделей, підбір параметрів та гіперпараметрів, тощо. Її задача – обробити сирий аудіо файл (або декілька файлів) за один прохід. Отже, уся складність розробки «схована» у одноразовому попередньому аналіз, тобто підготовці системи таким чином, щоб вона максимально швидко і результативно обробляла лише вхідні дані, проте при необхідності була здатна легко адаптуватись під нові вимоги (наприклад, додавання нового формату файлів, розширення датасету, поява нових жанрів, додавання нових моделей у ансамбль).

Загальний план підготовки та аналізу:

1) Збір даних;

2) Очистка даних (підготовка датасету в зручному для виділення ознак, навчання моделей та аналізу на етапі лише створення)

3) Виділення ознак (числових та візуальних);

4) Відбір ознак та підготовка даних;

5) Конфігурація (дизайн) моделей (створення, тренування, підбір параметрів, тестування);

6) Оцінка результатів, збереження найкращого ансамблю;

7) Демонстрація результатів.

3.1. Збір даних

Для того, щоб система була універсальною, тобто могла класифікувати музику незалежно від конкретних особливостей датасетів, було прийнято рішення використовувати їх комбінацію.

Набір даних GTZAN - це архів аудіо файлів формата .au, складається з 1000 аудіозаписів кожна 30 секунд. Він містить 10 жанрів, кожен

представлений по 100 треків. Доріжки - це 16-бітні моно-канальні аудіофайли 22050 Гц. При цьому кожен аудіо файл помічений відповідним класом завдяки розташуванні у відповідній директорії, а також назвою файлу у форматі disco.00005.au.

Free Music Archive (FMA) – незбалансований архів 100 тис. аудіозаписів, що розділені між 160 піджанрами. Доріжки будуть закодовані у форматі mp3, більшість із яких мають частоту дискретизації 44,100 Гц, швидкість передачі бітів 320 кбіт / с (в середньому 263 кбіт / с) та стерео-звук. Файли архіву FMA збережені неструктурно, названі відповідно до їх порядкового номеру в архіві. Усі метадані (розмітка жанрів) зберена окремо у вигляді .csv файлів.

Власна бібліотека представлена у вигляді директорії, що містить близько 1000 файлів, розміщених у піддиректоріях, що названі відповідно до жанру файлів. Аудіо форматів wav, mp3, різноманітні частота дискретизації, швидкість передачі бітів, кількість каналів (стерео-, моно-), жанри незбалансовані.

Для зручного збору даних було створено утиліту DataCollector.py, що зберігає усі файли у єдиному структурованому вигляді. DataCollector.py почергово запускає процеси організації файлів з різних джерел в унітарну структуру. Усі такі менеджери, що працюють зі структурую, мають спільний інтерфейс, отже утиліта може бути легко розширена для обробки інших датасетів, що мають інші правила файлової організації.

3.2. Очистка даних

Так як отримані при зборі даних жанри не обмежені та не збалансовані за кількістю екземплярів, необхідно враховувати, що до кожного жанру повинна відноситись велика кількість музичних файлів (хоча б більше 100), щоб класифікатори змогли виявити закономірності та розділити їх для різних жанрів. Проте результуючий набір даних також має бути більш-менш

збалансованим, щоб кількість файлів різних класів була однаковою, адже у противному випадку класифікаторам може бути вигідно в якості результату прогнозувати класи, що найбільше представлені у датасеті. Розподіл частоти потрапляння у неочищеному наборі даних зображено на рис. 3.1.

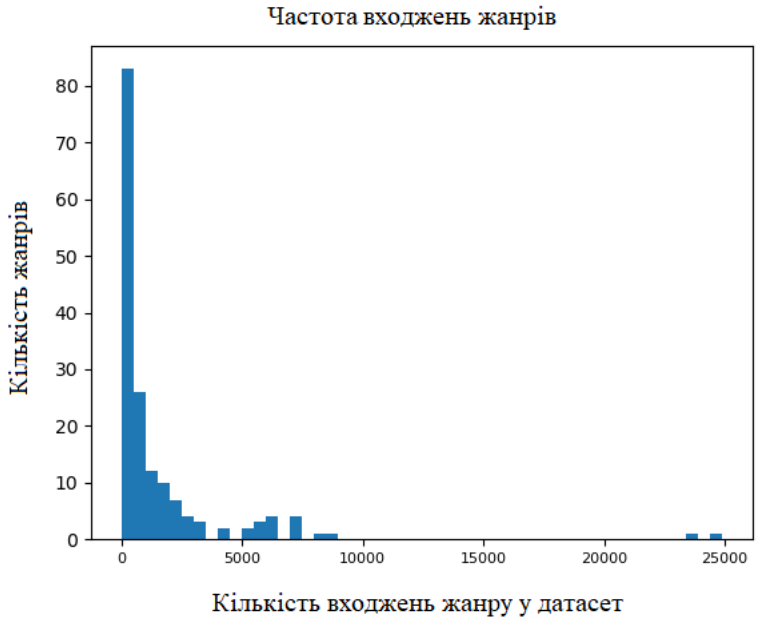


Рис 3.1 Частота потрапляння жанрів у неочищеному наборі даних

Також потрібно абстрагуватися від піджанрів і привести мітки файлів до батьківського жанру (супер-жанру), приклад яких відображений на рис. 3.2:

Рис. 3.2 Узагальнення піджанрів

Також для зручності подальшої обробки на етапі очистки даних необхідно конвертувати усі файли в один формат, такий щоб міг бути представлений і у вигляді спектрограми, і в вигляді числових ознак. Таким універсальним форматом є .wav. Файл складається з двох частин: заголовку файлу і області даних. Канонічний формат WAVE файлу починається з RIFF заголовку і двох підсекцій: "fmt" і "data". Підсекція "fmt" описує параметри даних звукозапису. У підсекції даних "data" міститься розмір даних і фактичні дані звукозапису. Структура файлу wav детально описана у таблиці 3.1:

Позиція (HEX)	Позиція	Розмір	Назва	Пояснення
• **RIFF Заголовок**				
0000	0	4 байти	ChunkID	"RIFF" в ASCII
0004	4	4 байти	ChunkSize	36 + SubChunk2Size, або 4 + (8 + SubChunk1Size) + (8 + SubChunk2Size) Це розмір всього файлу в байтах, без ChunkID і ChunkSize.
0008	8	4 байти	Format	"WAVE"
• **Підсекція "fmt "**				
0000C	12	4 байти	Subchunk1ID	“fmt "
00014	20	2 байта	AudioFormat	PCM = 1 (Лінійне квантування) Значення відмнінні від 1 вказують на наявність конкретного кодування аудіо даних.
00016	22	2 байта	NumChannels	Кількість звукових каналів. Моно = 1, Стерео = 2, і т.д.
00018	24	4 байти	SampleRate	Частота дискретизації у Гц
0001C	28	4 байти	ByteRate	SampleRate * NumChannels * BitsPerSample/8
00020	32	2 байта	BlockAlign	NumChannels * BitsPerSample/8 Кількість байт, яка міститься в одному семплі.
00022	34	2 байта	BitsPerSample	Кількість біт в одному семплі. Так звана "глибина" чи точність звучання. 8, 16, 32, і так далі.
		2 байта	ExtraParamSize	Розмір поля з параметрами. Якщо PCM, ці два поля ExtraParamSize і ExtraParams не записуються.
		X	ExtraParams	Місце для запису додаткових параметрів

Табл. 3.1 Структура WAV файлу

• **Підсекція "data"**				
00024	36	4 байти	Subchunk2ID	"data"
00028	40	4 байти	Subchunk2Size	Розмір даних звукозапису в байтах, які містяться в наступному полі. Кількість семплів * NumChannels * BitsPerSample/8
0002C	44	X	Data	Фактичні дані звукозапису.

Табл. 3.1 Структура WAV файлу

Також попередня очистка даних виконує конвертацію стереозвуку у моно у шляхом усереднення значень з каналів:

audiodata = audiodata.astype(float)

audiodata = audiodata.sum(axis=1) / 2

Очистка даних виконується утилітою DataCleaner.py.

3.3. Виділення ознак

Результуючий набір даних складається з 17000 .wav аудіозаписів, розподілених між 14 жанрами (класична, електронна, хіп-хоп, поп, рок, метал, інструментальна, диско, джаз, блюз, кантрі, фолк, експериментальна, реггі). Кожен з них збережений у піддиректорії, що названа відповідно до жанру.

Сирі .wav, навіть попередньо оброблені, не можуть бути вхідними даними для класифікаторів. Тому згідно описаних ознак в розділі 2.2. кожен файл повинен бути представлений у вигляді таких його характеристик (числових або візуальних ознак):

1) Мел-частотні кепстральні коефіцієнти (MFCC);

2) Спектральний центроїд;

3) Частота перетину нуля;

4) Частота кольоровості;

5) Спектральний спад частоти;

6) Спектрограма (зображення).

Для виділення таких ознак використано бібліотеку librosa. Librosa може

працювати з будь-якими звуковими сигналами, але орієнтована в основному саме на музику. Вона дозволяє створити повноцінну систему вилучення музичної інформації (MIR).

librosa.load() перетворює аудіо формат .wav у масив numpy, що є стандартом представлення даних для задач машинного і глибинного навчання.

Числові ознаки, що можна отримати, використовуючи функції бібліотеки librosa:

Функція librosa	Характеристика (ознака)
librosa.feature.zero_crossing_rate()	Частота перетину нуля
librosa.feature.spectral_centroid()	Спектральний центроїд
librosa.feature.spectral_rolloff()	Спектральний спад частоти
librosa.feature.chroma_stft()	Частота кольоровості
librosa.feature.mfcc()	Мел-частотні кепстральні коефіцієнти (20 штук)

Табл. 3.2 Використані функції бібліотеки librosa

Для представлення аудіо у вигляді зображень(спектрограм) також використано функціональність librosa, а саме функцію librosa.feature.melspectrogram(). За допомогою librosa.power_to_db() перетворили вісь у логарифмічну (усі явні відмінності знаходяться у нижній частину спектру, а отже і зображень). Використано pylab.savefig() та можливості бібліотеки matplotlib для зберігання отриманого результату у форматі jpeg.

Отже, в результаті виділення ознак:

1) Створено набір даних, представлених 518 числовими ознаками, збережений у .csv форматі;

2) Створено набір даних, представлених відображенням аудіо зображеннями у форматі .jpg.

3.4. Відбір ознак та підготовка даних

Для зменшення розмірності числового набору даних було застосовано метод головних компонент PCA. Такий підхід дозволив суттєво зменшити час обробки даних, майже не втрачаючи в кількості інформації, що зберігається. Бібліотека sklearn ховає складність методу дозволяє використовувати PCA дуже просто:

```
from sklearn.decomposition import PCA
pca = PCA(n_components).fit(X)
X = pca.transform(X)
```

Дослідним шляхом встановлено, що доцільно використовувати 150-250 компонент, без ризику втрати повноту інформмації (рис 3.3). Тобто розмірність зменшилась в 2-3 рази, що відобразилось в швидкості навчання моделей.

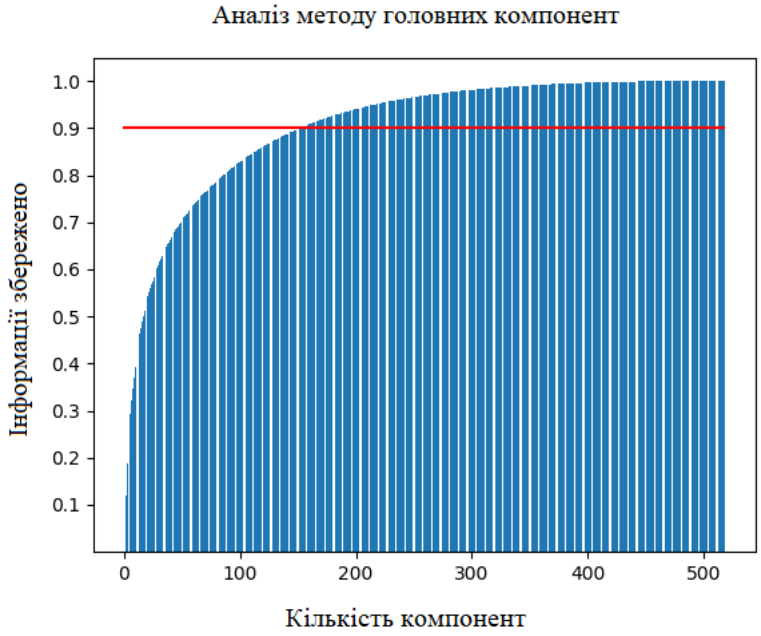


Рис 3.3 Аналіз ефективності PCA

Для покращення ефективності роботи системи результуючий набір даних розділяється на три групи, що використовуються на різних етапах створення моделей:

• тренувальний набір (70%). Використовується безпосередньо для навчання моделей (корегування параметрів моделі);

• валідаційний (затверджувальний) набір (20%). Забезпечує об'єктивну оцінку моделей, яка відповідає навчальному набору при налаштуванні гіперпараметрів моделей (наприклад, число прихованих шарів нейронної мережі). Також використовується для регуляризації шляхом ранньої зупинки: навчання переривається, коли помилка на наборі даних для затверджування збільшується, оскільки це є ознакою перенавчання на навчальному наборі даних;

• тестовий набір (10%). Прилади не перекриваються з тренувальною вибіркою. Використовується для забезпечення об'єктивної оцінки кінцевої моделі

Розподіл відбувається випадковим чином, але шляхом стратифікованих проб. Тобто розподіл по жанрам у кожному з наборів відповідає такому розподілу цілого набору.

LabelEncoder

StandardScaler

Slices

Scikit-learn — це безплатна бібліотека машинного навчання, написана на Python. Вона надає широкий вибір алгоритмів навчання з учителем і без нього. Одна з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек і легко інтегрує їх одна з одною. Ще однією перевагою є широка спільнота й докладна документація. Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення задач навчання з учителем: класифікації й регресії, а також для завдань навчання без учителя: кластеризації, зменшення розмірності й детектування аномалій.

3.5. Створення, тренування і налаштування класифікаторів

3.5.1 Випадковий ліс

Реалізація алгоритму випадкового лісу була виконана за допомогою бібліотеки Scikit-Learn:

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

Налаштування класифікатора полягає у визначенні його гіперпараметра n_estimators (кількість дерев). Дослідницьким шляхом визначено якість алгоритму залежно від кількості дерев (рис 3.4):

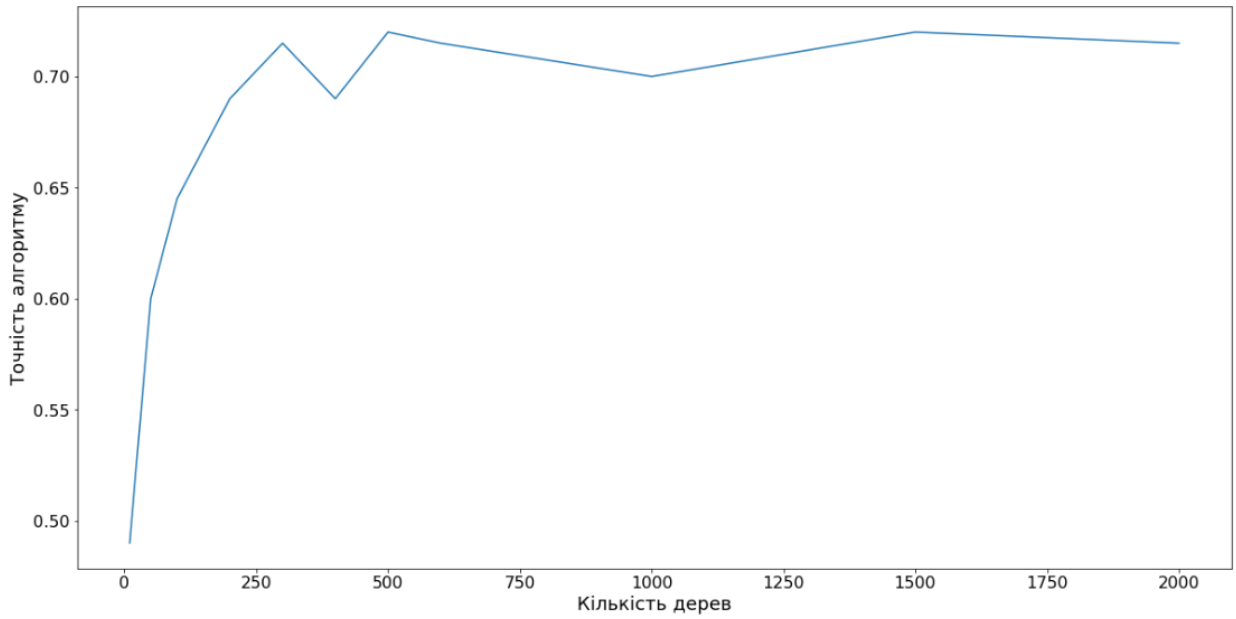


Рис. 3.4 Залежність точності алгоритму Random Forest Classifier від гіперпараметра n_estimators

Зроблено висновок, що 500 – найбліш вдала кількість дерев у лісі. Натренований на 500 деревах алгоритм випадкового лісу показав такі результати:

Test Set Accuracy = 0.70

Test Set F-score = 0.70

ROC AUC = 0.959

Матриця плутанини(рис. 3.5):

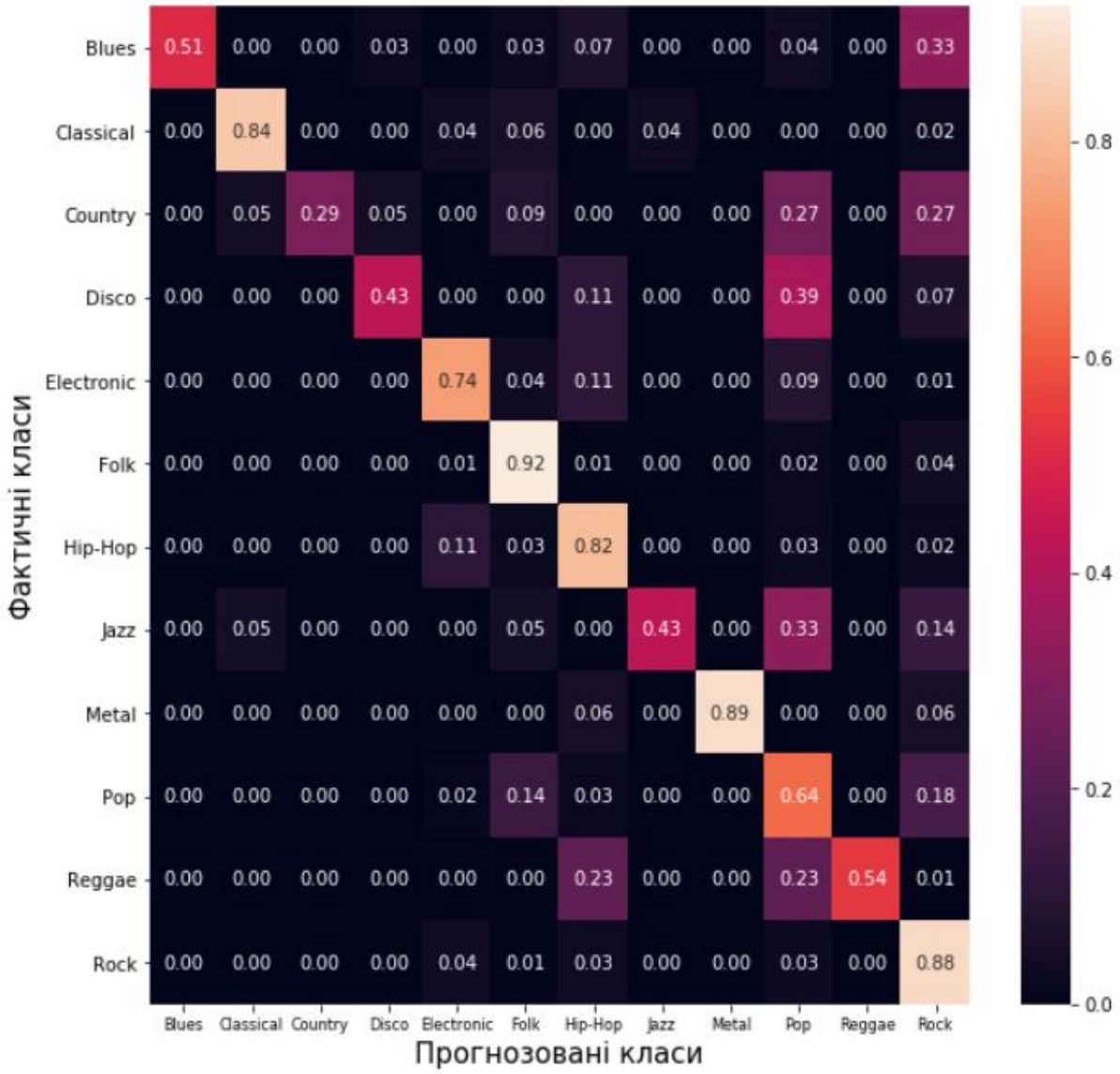


Рис. 3.5 Матриця плутанини для оцінки роботи Random Forest на тестовому наборі даних

Отже, досить непогано алгоритм визначає Classical, Folk, Metal, Rock, але робить багато помилок на Country, Disco, Jazz, Blues, Reggae: дійсно, Disco часто можна переплутати з Pop музикою, кантрі – з Pop або Rock, а реггі це і є щось середнє між Pop і Hip-Hop.

3.5.2. Класична глибинна мережа

Реалізація алгоритму випадкового лісу була виконана за допомогою бібліотеки Keras, що працює поверх TensorFlow. Бібліотека TensorFlow — це потужна програмна бібліотека з відкритим кодом, розроблена компанією Google. Вона дуже добре підходить і точно підігнана під великомасштабне машинне навчання. Кожне обчислення в TensorFlow представляється як граф потоку даних, він же граф обчислень. Граф обчислень є моделлю, яка описує як будуть виконуватися обчислення. Граф визначається користувачем в Python, а TensorFlow виконує кожну операцію над графом, ядра яких реалізовано на C++ з використанням бібліотек Eigen і cuDNN для кращої продуктивності. Найголовніше, граф можна розбивати на частини й проганяти їх паралельно на безлічі центральних процесорів або графічних процесорів. Бібліотека TensorFlow також підтримує розподілені обчислення по різним серверам, тому є можливість навчати величезні графи на гігантських навчальних наборах швидко.

Налаштування класифікатора полягає у визначенні таких його параметрів та гіперпараметрів:

1) кількість шарів та нейронів в кожному;
2) функція активації нейронів;
3) оптимізатор;
4) кількість епох та batch розмір;
5) вірогідність виключення нейронів;
6) початкова ініціалізація нейронів.

Було запущено пошук найкращих таких характеристик методом повного перебору по сітці. В результаті отримано декілька наборів параметрів, що на валідаційному датасеті дали однаковий кращий результат. Однак за загальнонауковим принципом «чим простіше, тим ліпше», а також за припущенням, що різниця між точністю на валідаційному і навчальному

наборах має бути чим менше (в протилежному випадку велика різниця може свідчити про перенавчання) було обрано таку модель, що:

1) Вхідний шар містить 200 нейронів відповідно до кількості відібраних ознак. Далі три прихованих шари по 256, 128 і 64 нейрони, функція активації кожного – ReLU. Початкова ініціалізація – за методом Завієра;

2) Вихідний шар містить 14 нейронів, що відповідає кількості жанрів. Функція активації вихідного шару – softmax;

3) Оптимізатор – Adam;

4) Кількість епох – 20, batch size – 64;

5) Вірогідність дропауту – 0.2;

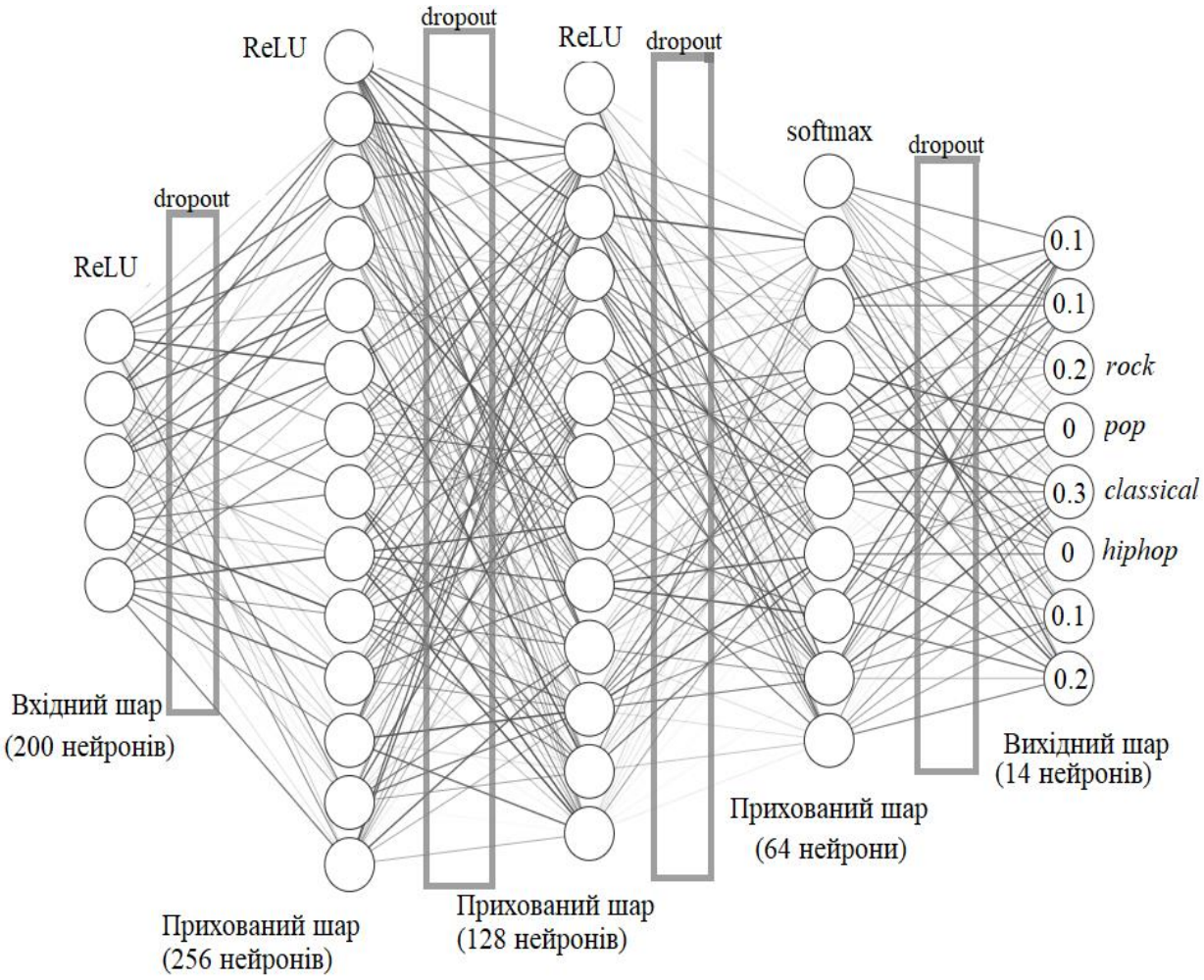


Рис. 3.6 Архітектура нейронної мережі

Фрагмент основного коду створення і налаштування остаточної моделі за допомогою Keras зображено на рис 3.7.

```
from keras import models
from keras import layers


dropout_proba = 0.2
epochs = 20
batch_size = 64


model = models.Sequential()

model.add(layers.Dense(256, activation='relu', kernel_initializer='he_normal', input_shape=(X_train.shape[1],)))
model.add(layers.Dropout(dropout_proba))

model.add(layers.Dense(128, activation='relu', kernel_initializer='he_normal'))
model.add(layers.Dropout(dropout_proba))

model.add(layers.Dense(64, activation='relu', kernel_initializer='he_normal'))
model.add(layers.Dropout(dropout_proba))

model.add(layers.Dense(len(Counter(y)), activation='softmax', kernel_initializer='he_normal'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


history = model.fit(X_train,
                    y_train,
                    epochs=epochs,
                    batch_size=batch_size, verbose = 1)
```

Рис. 3.7 Фрагмент коду Python, налаштування моделі нейромережі за допомогою Keras

Натренована модель показала такі результати на тестовому наборі даних:

Test Set Accuracy = 0.80

Test Set F-score = 0.84

ROC AUC = 0.962

Оскільки за всіма метриками якості модель нейронної мережі суттєво переважає над результатами алгоритму випадкового лісу, доцільність поєднання таких алгоритмів в ансамбль викликає сумніви, адже за умови рівноправного голосування гірший з алгоритмів обов'язково знизить ефективність ансамблю до рівня, що буде навіть нижче ніж кращий алгоритм демонструватиме «самотужки».

Такий висновок значно спрощує складність системи, адже зникає необхідність не лише калібрування ймовірностей і голосування для Random

Forest на рівні одного файлу, а й відпадають такі ж операції на рівні одного типу ознак (числового) (рис 2.17)

Матриця плутанини найращої нейронної мережі зображена на рис 3.8.

Мережа в цілому справляється з задачею класифікації помітно краще, ніж RF, проте Reggae все ще інколи прогнозується як Hip-Hop, а Pop істотно програє у точності виявлення іншим жанрам, можливо через різноманітність та неоднозначність жанру.

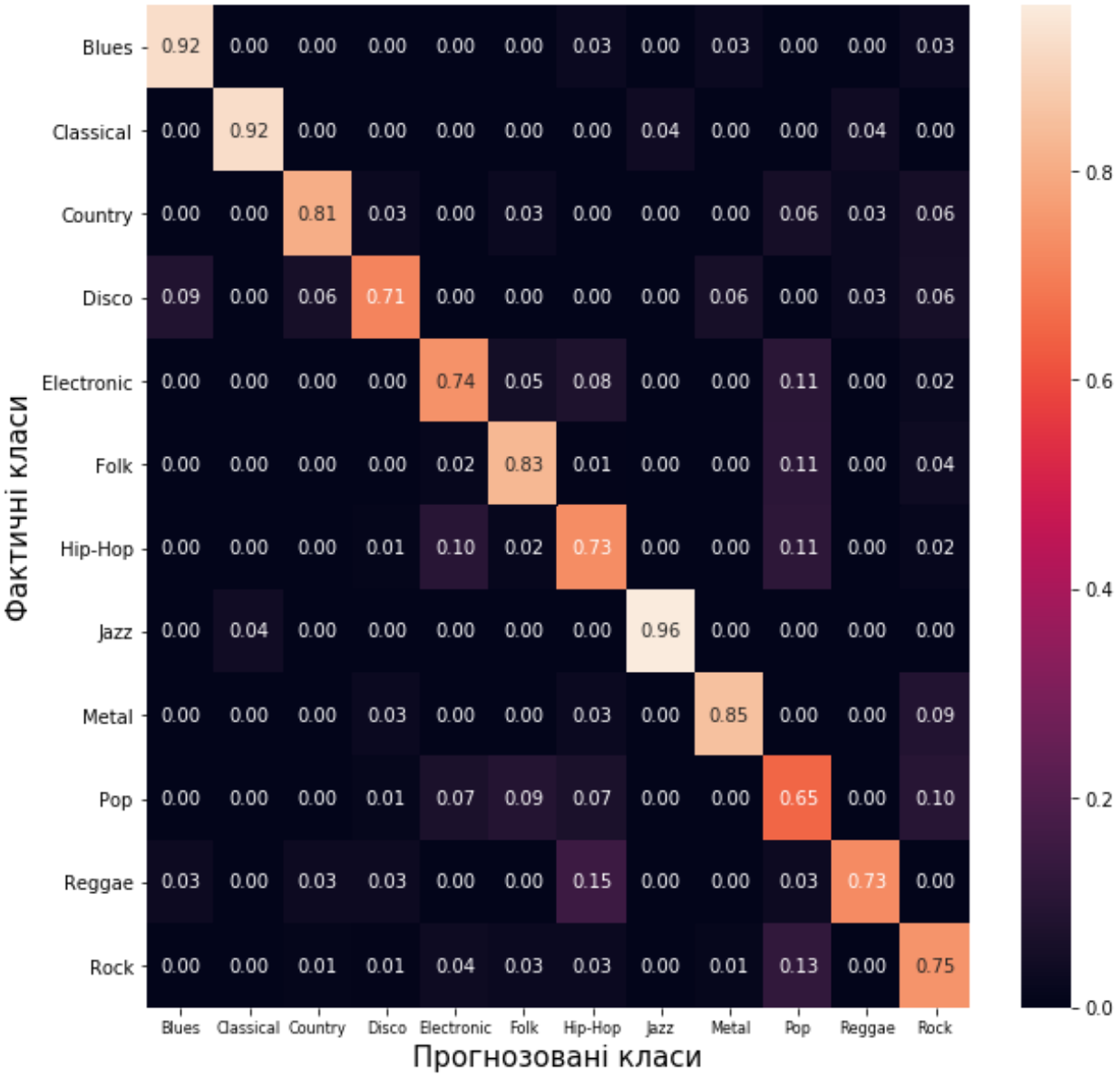


Рис. 3.8 Матриця плутанини для оцінки роботи FF-DNN на тестовому наборі даних

3.5.3. Згорткова нейронна мережа

Архітектура мережі з параметрами її налаштування була запозичена з модифікацією у одного з state-of-the-art рішень [12], її схематично зображено на рис 3.9.

1) Вхідний шар містить 128x128x3 нейронів відповідно до кількості розміру нарізаних частин спектрограм (128x128), 3 – кількість каналів кольорів;

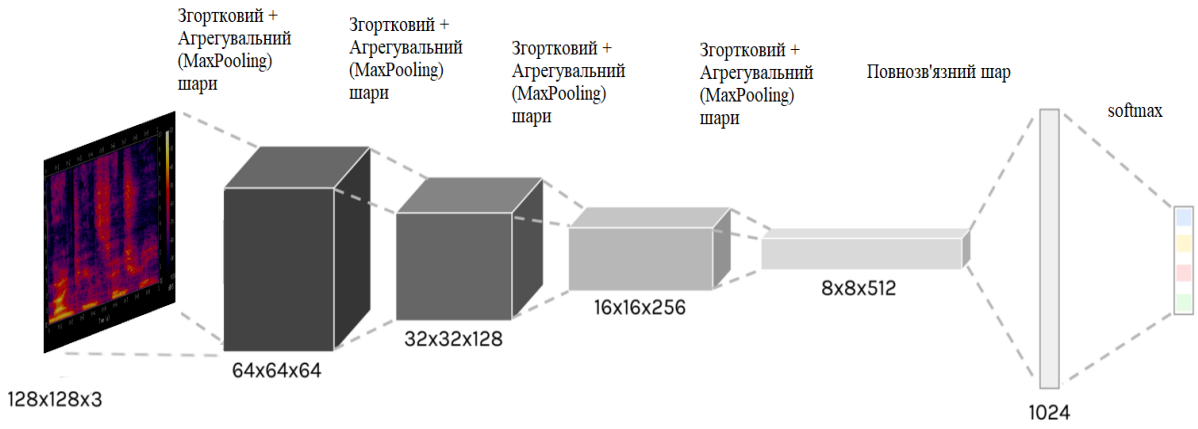


Рис. 3.9 Архітектура згорткової мережі

2) Вихідний шар містить 14 нейронів, що відповідає кількості жанрів. Функція активації вихідного шару – softmax;

3) Функція активації – ReLU.

4) Початкова ініціалізація усіх ваг мережі – за методом Завієра;

3) Оптимізатор – RMSProp;

4) Вірогідність дропауту – 0.3;

Для цього класифікатора час навчання набагато більший за час попередніх алгоритмів, адже кожен семпл – тривимірна матриця (на відміну від попередніх одновимірних векторів), а архітектура набагато складніша. Тому підбір параметрів та гіперпараметрів такої моделі шукати методом повного перебору було б занадто довготривалим процесом. Отже така

конфігурація є найкращою лише умовно, є великий простір для покращення ефективності саме цієї моделі, адже до того архітектури згорткових мереж ще й часто здобувають покращення разом з ускладненням, чого не можна сказати про класичну глибинну нейронну мережу, де найкращі результати зазвичай можна отримати з найпростішою архітектурою. Реалізація алгоритму випадкового лісу була виконана за допомогою бібліотеки TFLearn, що працює поверх TensorFlow, подібно до Keras зображена нижче:

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

cnn = input_data(shape=[None, imageSize, imageSize, 3], name='input')

cnn = conv_2d(cnn, 64, 2, activation='elu', weights_init="Xavier")
cnn = max_pool_2d(cnn, 2)

cnn = conv_2d(cnn, 128, 2, activation='elu', weights_init="Xavier")
cnn = max_pool_2d(cnn, 2)

cnn = conv_2d(cnn, 256, 2, activation='elu', weights_init="Xavier")
cnn = max_pool_2d(cnn, 2)

cnn = conv_2d(cnn, 512, 2, activation='elu', weights_init="Xavier")
cnn = max_pool_2d(cnn, 2)

cnn = fully_connected(cnn, 1024, activation='elu')
cnn = dropout(cnn, 0.3)

cnn = fully_connected(cnn, nbClasses, activation='softmax')
cnn = regression(cnn, optimizer='rmsprop', loss='sparse_categorical_crossentropy')

model = tflearn.DNN(cnn)
```

Рис. 3.10 Фрагмент коду Python, налаштування моделі згорткової нейромережі за допомогою TFLearn

Натренована згорткова мережа показала такі результати на тестовому наборі даних:

Test Set Accuracy = 0.82

Test Set F-score = 0.84

AUC ROC = 0.970

Матриця плутанини зображена на рис. 3.5, де помітно, що результати роботи згорткової мережі дуже подібні до результатів моделі з розділу 3.5.2, а отже доцільно спробувати використати створити ансамбль голосування саме

на основі CNN і DNN, за умови попереднього застосування до результатів калібрування.

Калібрування обох моделей виконано засобами sklearn, а саме «огортанням» моделей ще до початку навчання у sklearn.calibration.CalibratedClassifierCV

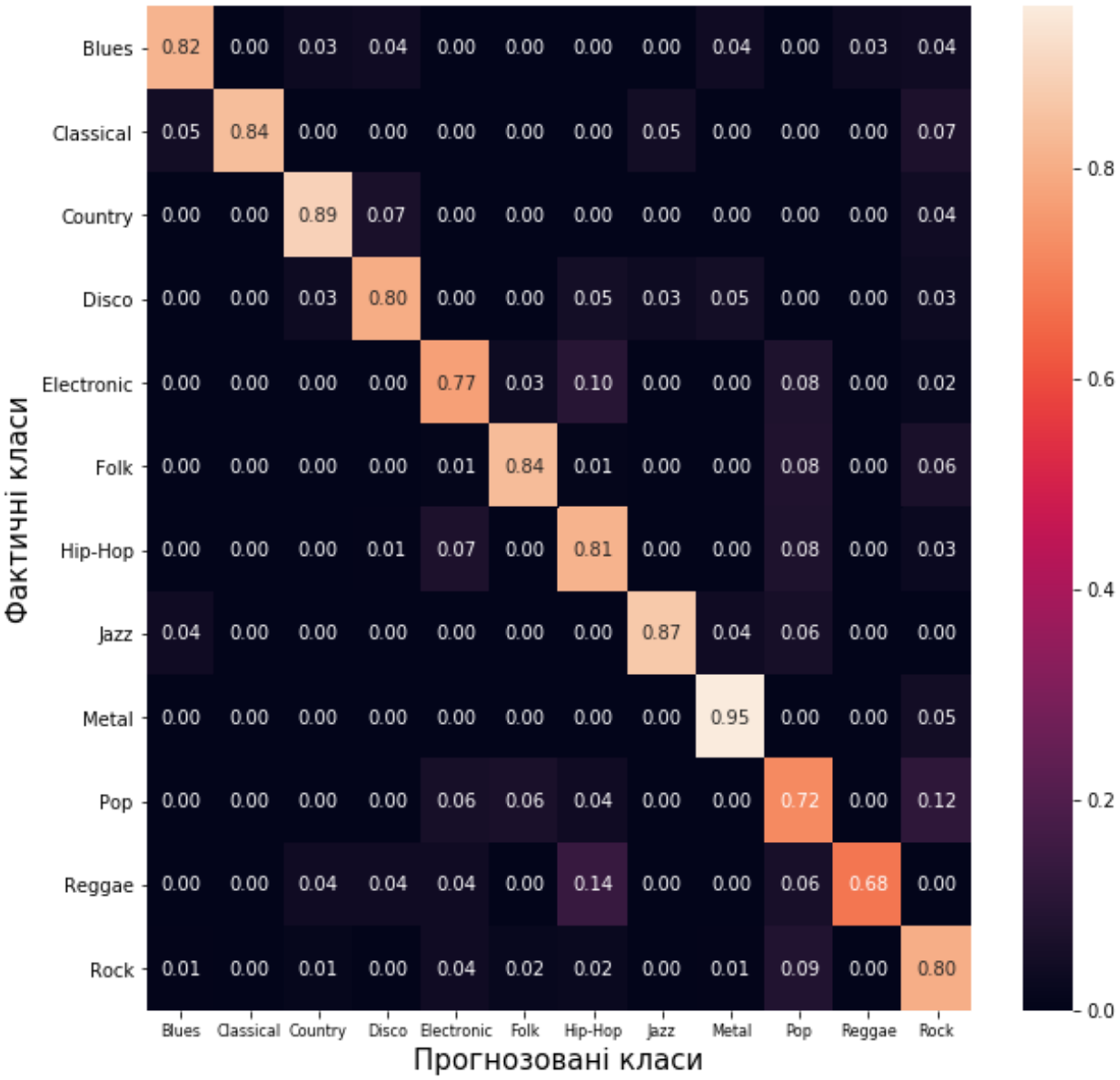


Рис. 3.5 Матриця плутанини для CNN на тестовому наборі даних

3.6. Оцінка результатів

Порівняння отриманих метрик якості моделей на тестовому наборі даних наведено у табл. 3.3:

	Accuracy	F-score	AUC-ROC
Моделі натреновані на числових ознаках			
Random Forest	**0.70**	**0.70**	**0.959**
DNN	0.80	**0.84**	0.962
Моделі натреновані на графічних ознаках			
CNN	0.82	**0.84**	0.970
Ансамблі моделей			
CNN + DNN	**0.83**	**0.84**	**0.971**

Табл. 3.3 Порівняння метрик алгоритмів класифікації

Отримані результати можна інтерпретувати так: класифікатори справляються з задачею, але алгоритм Випадкового лісу набагато пунктів відстає від інших за усіма метриками, отже його присутність у системі у процесі рівноправного голосування буде шкодити загальній ефективності. Згорткова нейронна мережа працює в цілому краще ніж класична глибинна, але їх поєднання у голосуючий ансамбль несуттєво, але таки покращує якість системи за усіма трьома досліджуваними метриками. Чисельні значення метрик для усіх трьох якісно різних методів глибокого навчання та їх комбінації (DNN, CNN, DNN + CNN) демонструють незначний розкид значень, що дозволяє оцінити похибку метрик класифікації (в межах 0.01). Однак при пошуку точок вдосконалення системи на даному наборі даних доцільно концентруватись на таких підходах:

1. покращенні згорткової нейронної мережі і використання тільки її у системі. Адже, як було описано раніше, глибинна мережа навряд чи може бути значно вдосконалена, на відміну від згорткової. Такий підхід також зменшить складність системи за рахунок видалення частин обробки та підготовки даних, налаштування моделей, ансамблевого голосування і калібрування

ймовірностей;

2. зміна методу голосування. Наприклад, замість використання рівноправного голосування можна використовувати зважене, тоді більш вдалим алгоритмам буде додано більше ваги у голосуванні;

3. використання алгоритму підсилювання. З таким підходом алгоритми будуть більше зосереджуватись на тих прикладах, які попередні слабкі алгоритми(Випадковий ліс) класифікували неправильно

4. дослідження інших алгоритмів класифікації

ВИСНОВКИ ДО РОЗДІЛУ

При розробці системи було створено зібрано великий набір різноманітних аудіо файлів 14 жанрів, виділено і підготовлено графічні та числові ознаки, що були використані у класифікації різними моделями, зібрано класифікатори в ансамбль.

Результати використання моделей Random Forest, Feedforward Deep Neural Network, Convolutional Neural Network цілком задовольняє вимоги до вирішення поставленої задачі класифікації у системі за усіма трьома досліджуваними метриками. Чисельні значення метрик для усіх трьох якісно різних методів глибокого навчання та їх комбінації (DNN, CNN, ансамбль DNN + CNN) демонструють незначний розкид значень, що дозволяє оцінити похибку метрик класифікації (в межах 0.01).

На відміну від існуючих рішень отримані результати демонструють високу стабільність отриманих значень метрик, яка підтверджується незначним розкидом значень для окремих запропонованих моделей та їх ансамблю, а також невеликою розбіжністю між значеннями їх метрик на тестововому, валідаційному і тренувальному наборах, це свідчить й про те, що можливе вдосконалення шляхом поповнення кількості тренувальних даних.

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання дипломного проекту була розроблена система, яка вирішує проблему класифікації аудіо контенту, а також зібрано та підготовлено новий набір даних, що може використовуватись у задачах з області пошуку музичної інформації та її аналізу.

Були переглянуті та проаналізовані існуючі рішення, вказані їх недоліки та переваги. Така система за метриками якості працює не гірше усіх рішень (з 84% точністю визначає жанр) за всіма трьома досліджувальними метриками, що вже існують сьогодні, а її новизна і перевага над ними – репрезентативний нестандартизований набір даних, використання якого свідчить про універсальність системи, кількість жанрів для класифікації (14 проти популярних 10), незначний розкид значень для окремих запропонованих моделей та їх ансамблю, а також невелика розбіжністю між значеннями їх метрик на тестововому, валідаційному і тренувальному наборах (ознака перенавчання).

Система також передбачає можливість пакетної класифікації, переналаштування на нові дані та візуалізацій результатів, додавання нових даних.

Крім того, натреновані на класифікації жанрів та підготовлені для цього дані можна використати як основу для більш широких задач, наприклад пошуку схожих музичних композицій або music summarization.

Були описані алгоритми та реалізації необхідої для системи функціональності, такі як: обробка аудіо файлів різних форматів, виділення числових та графічних ознак для класифікації та алгоритм їх відбіру ознак, створення і налаштування класифікаторів за допомогою концепцій машинного навчання та глибинного навчання з використанням різних типів нейромерж (Випадковий ліс, глибинна нейронна мережа, згорткова нейронна мережа), поєднання класифікаторів у ансабль.

Для реалізації проекту була вибрана мова програмування, яка найбільш відповідала вимогам реалізації – Python. Використано такі основні бібліотеки та фреймворки: TensorFlow, Keras, pandas, numpy, Scikit-learn librosa, matplotlib, seaborn. Проект розроблявся з використанням Jupyter notebook.

ПЕРЕЛІК ПОСИЛАНЬ

1. AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources. – Режим доступу до ресурсу: https://arxiv.org/pdf/1911.12618.pdf

2. Brazilian Lyrics-Based Music Genre Classification Using a BLSTM Network [Електронний ресурс] – Arxiv, 2020. – Режим доступу до ресурсу: https://arxiv.org/pdf/2003.05377.pdf

3. Birla M. Partial run-time reconfiguration of FPGA for computer vision applications / M. Birla, K. N. Vikram. // Parallel and Distributed Processing, 2008 (IPDPS 2008). (Miami, April 14—18, 2008). — IEEE Computer Society, 2008. — P. 1—6.

4. Combining CNN and Classical Algorithms for Music Genre Classification [Електронний ресурс] – Режим доступу до ресурсу:

http://cs229.stanford.edu/proj2018/report/19.pdf

5. Audio spectrogram representationsfor processing with convolutional neural networks [Електронний ресурс] – Arxiv, 2020. – Режим доступу до ресурсу: https://arxiv.org/pdf/1706.09559.pdf

6. Nanni, Loris & Costa, Yandre & Lumini, Alessandra & Kim, Moo & Baek, Seung. (2015). Combining visual and acoustic features for music genre classification. Expert Systems with Applications.

7. Сирота Р. Б. Проблеми побудови каліброваних систем/ Р. Б. Сирота,

Д. Я. Ширин // Методы и алгоритмы анализа данных и их моделирование в MATLAB, 2010. — № 7 (48). — С. 200—204.

5. Audio spectrogram representationsfor processing with convolutional neural networks [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1706.09559.pdf

8. A comparative analysis of CNN and LSTM for music genre classification [Електронний ресурс]. – Режим доступу до ресурсу: https://www.diva-portal.org/smash/get/diva2:1354738/FULLTEXT01.pdf

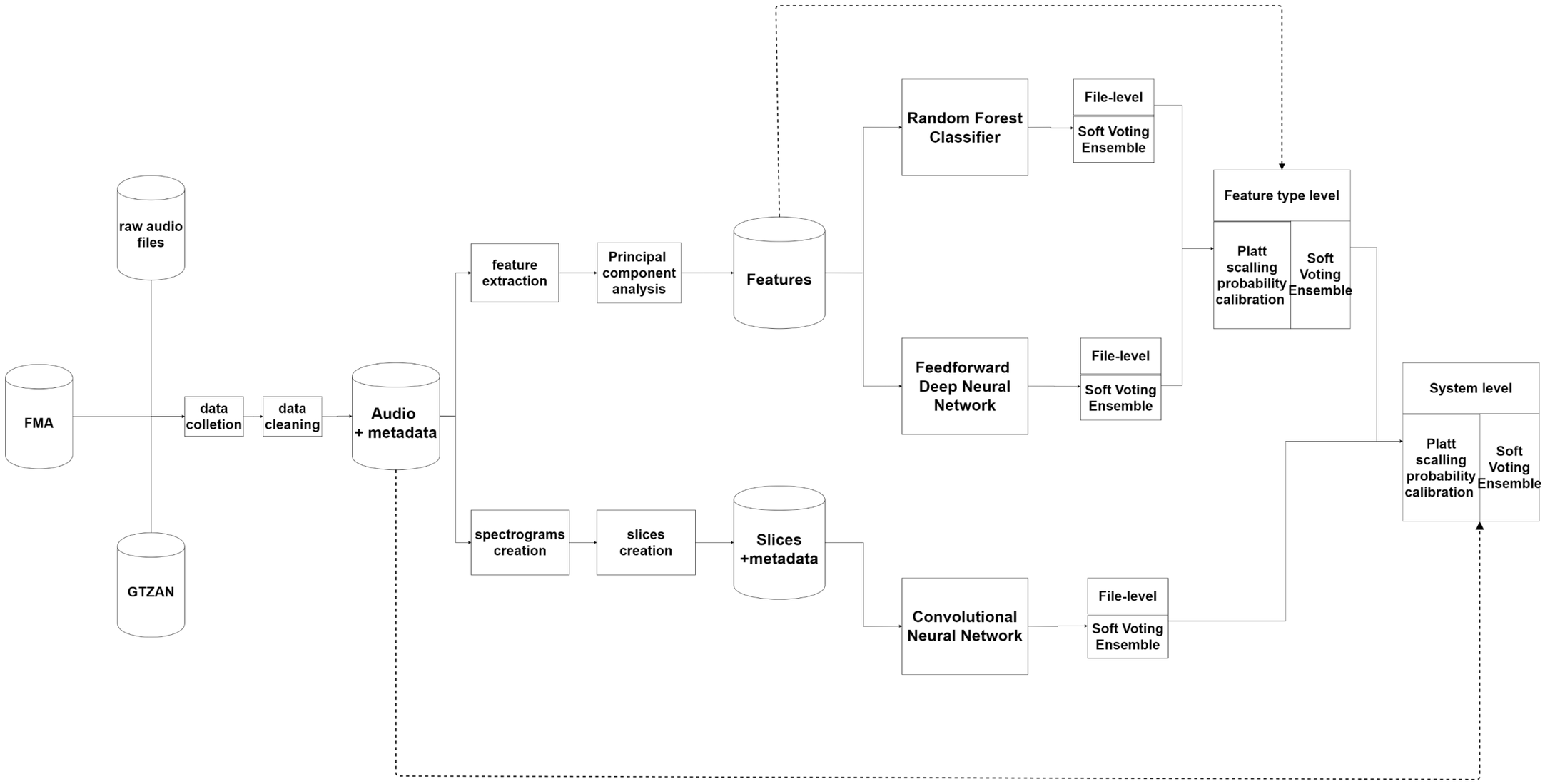
9. Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal Processing Magazine 29.6 (2012): 82-97.

10. Stanford U. CS231n: Convolutional Neural Networks for Visual Recognition/ Univercity Stanford – Режим доступу до ресурсу: http://cs231n.github.io/.

11. Смоленцев Н. Основы теории вейвлетов. Вейвлеты в Matlab [Текст] / Н. Смоленцев. – М.: ДМК Пресс 2014. – 628 с.

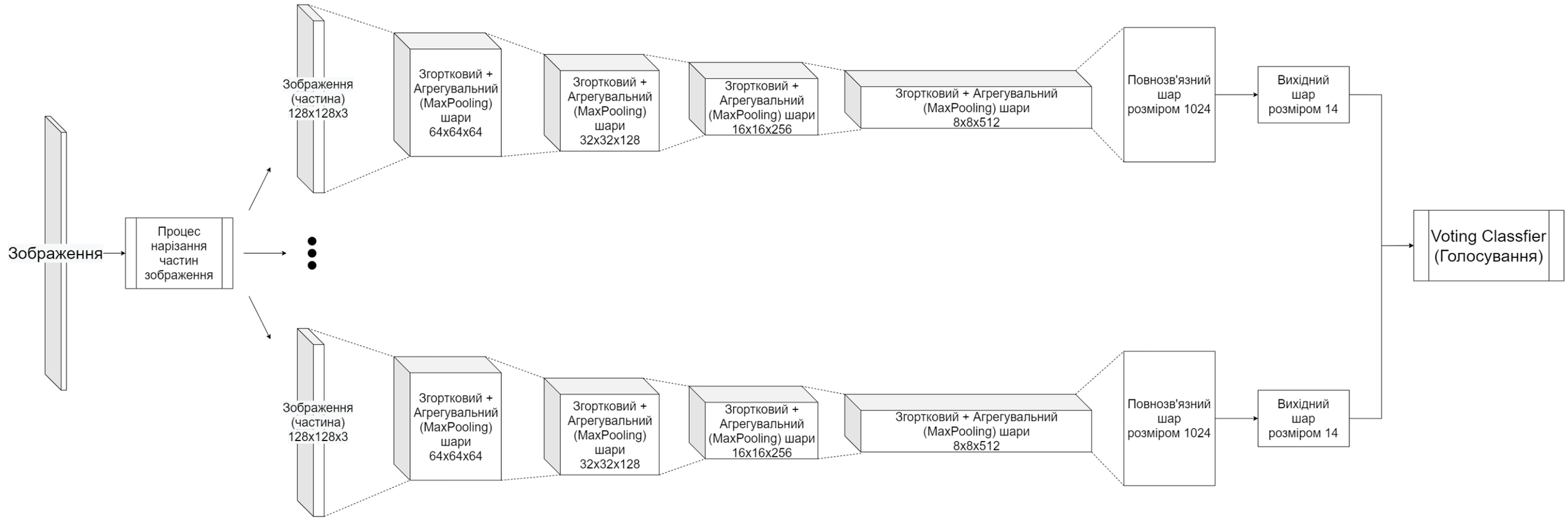
12. Music Genre Classification using Machine Learning Techniques [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1804.01149

13-15. Бинарная классификация жанров аудио на признаках Spotify [Електронний ресурс] – Режим доступу: https://github.com/reachanihere/Song-Genre-Classification



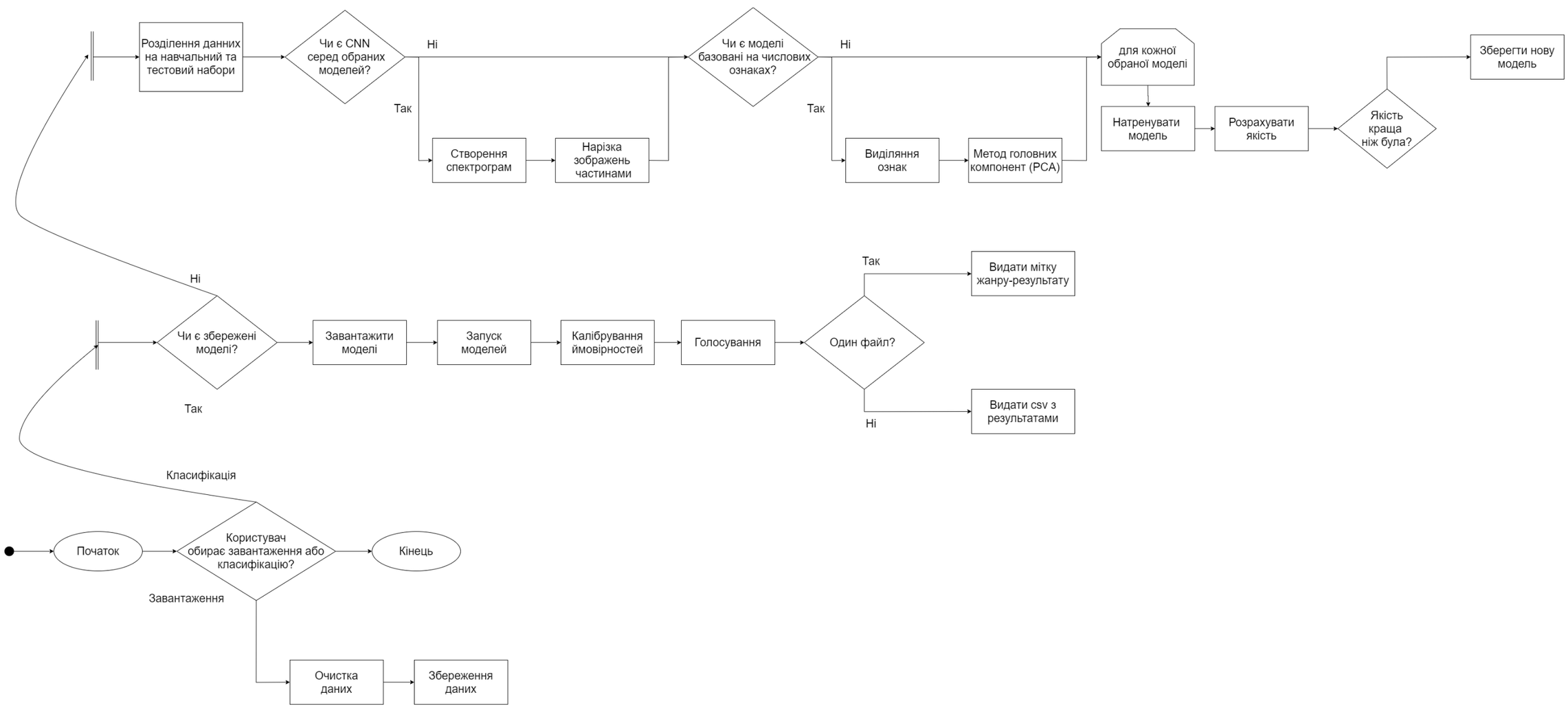
					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Принципова схема	Літ. / Маса / Масштаб
Розроб.		Валігура І.А.				
Перевір.		Гордієнко Ю.Г.				Аркуш 1 / Аркушів 1
Н. Контр.		Сімоненко В.П.			Дипломна робота	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62
Затв.						

Підпис і дата | Інв. № дубл. | Взам. інв. № | Підпис і дата | Інв. № оригі.



					ІАЛЦ.467100.005 Д2			
Зм.	Арк.	№ докум.	Підпис	Дата	Структурна схема класифікації зображень згортковою нейронною мережею	Літ.	Маса	Масштаб
Розроб.		Валігура І.А.						
Перевір.		Гордієнко Ю.Г.				Аркуш 1	Аркушів 1	
Н. Контр.		Сімоненко В.П.			Дипломна робота	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62		
Затв.								

Інв. № ориг. | Підпис і дата | Взам. інв. № | Інв. № дубл. | Підпис і дата



					ІАЛЦ.467100.006 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата	Функціональна схема	Літ.	Маса / Масштаб
Розроб.		Валігура І.А.					
Перевір.		Гордієнко Ю.Г.				Аркуш 1	Аркушів 1
Н. Контр.		Сімоненко В.П.			Дипломна робота	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62	
Затв.							

Підпис і дата | Інв. № дубл. | Взам. інв. № | Підпис і дата | Інв. № ориг.